普通高等教育“十三五”规划教材

# “蓝桥杯”
# 全国软件和信息技术专业人才大赛
# （电子类）实训指导书


彭大海　主编

蒋水秀　郭书军　副主编

朱青建　审核



電子工業出版社

**Publishing House of Electronics Industry**

北京・BEIJING



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

VISIT…

## 内 容 简 介

本书分别以“蓝桥杯”大赛单片机竞赛实训平台和嵌入式竞赛实训平台为硬件环境，在介绍实训平台模块程序设计的基础上，介绍单片机设计与开发项目和嵌入式设计与开发项目的试题解析。

全书分为 6 章，第 1 章为大赛简介与经验交流，第 2 章为单片机和嵌入式的开发环境与工具，第 3 章为单片机设计与开发项目模块程序设计，第 4 章为单片机设计与开发项目往届试题解析，第 5 章为嵌入式设计与开发项目模块程序设计，第 6 章为嵌入式设计与开发项目往届试题解析。书后附有竞赛实训平台简介和竞赛大纲。

本书所有设计程序均为原创，并经过多次实验反复改进，内容简单易懂，特别适合用于大赛的实训指导。本书可以作为单片机和嵌入式系统设计教材，供电子、通信和自动化等相关专业的学生使用。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。
版权所有，侵权必究。

**图书在版编目（CIP）数据**

“蓝桥杯”全国软件和信息技术专业人才大赛（电子类）实训指导书 / 彭大海主编. —北京：电子工业出版社，2019.11
ISBN 978-7-121-37655-9

Ⅰ. ①蓝… Ⅱ. ①彭… Ⅲ. ①软件开发－高等学校－教学参考资料②微型计算机－高等学校－教学参考资料
Ⅳ. ①TP311.52②TP36

中国版本图书馆 CIP 数据核字（2019）第 242314 号

责任编辑：赵玉山
印　　刷：
装　　订：
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编 100036
开　　本：787×1 092　1/16　印张：18　字数：460 千字
版　　次：2019 年 11 月第 1 版
印　　次：2019 年 11 月第 1 次印刷
定　　价：59.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：（010）88254556，zhaoys@phei.com.cn。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

# 前　言

为促进软件和信息领域专业技术人才培养，提升高校毕业生的就业竞争力，工业和信息化部人才交流中心主办了“蓝桥杯”大赛。截至 2019 年 6 月，“蓝桥杯”大赛已经连续举行了 10 届，包括清华大学、北京大学在内的超过 1200 所院校、30 万名选手报名参加了大赛，已成为具有重大社会影响力和行业认可度的 IT 类学科竞赛。

“蓝桥杯”大赛电子类竞赛目前开设单片机设计与开发和嵌入式设计与开发两个科目。单片机科目选用宏晶科技的增强型 51 单片机 IAP15F2K61S2，支持串口调试；嵌入式科目选用意法半导体 ARM Cortex M3 内核的中密度微控制器 STM32F103RBT6。大赛以封闭限时的方式考核选手单片机、嵌入式基础知识的掌握和基于硬件的程序系统设计及调试能力。

一直以来，“蓝桥杯”大赛电子类竞赛的参赛师生都希望组委会编撰一本专门针对大赛单片机设计与开发和嵌入式设计与开发两个科目的竞赛指导书。值此“蓝桥杯”成立十周年之际，大赛组委会特委托北京国信长天科技有限公司组织编写了本书，以利于指导电子类竞赛的参赛师生准备大赛。

本书结合“蓝桥杯”大赛电子类竞赛大纲要求，首先介绍了大赛采用的开发环境与工具，包括 Keil C51、Realview MDK、软件开发包及相关调试和下载工具，然后按照模块化的方式由易到难逐一介绍单片机和嵌入式竞赛实训平台上的硬件模块程序设计方法，最后结合历年电子类竞赛真题，按照系统设计、初始化程序设计、处理程序设计和系统实现的思路对真题进行详细解析。

在本书的编写过程中，“蓝桥杯”大赛组委会副秘书长彭大海先生对本书做了全面的统筹规划并编写了第 1 章，蒋水秀教授编写了 2.1 节、第 3 章和第 4 章等单片机科目相关内容，郭书军教授编写了 2.2 节、第 5 章和第 6 章等嵌入式科目相关内容并对全书进行了统稿，本书的审核由大赛组委会资深技术专家朱青建先生完成。此外，本书还特别组织了技术源厂商专家、用人单位专家和参赛获奖师生就单片机和嵌入式的学习及实践，从各自的角度撰写了经验交流文章，邀请沈孟锋老师、徐观生老师对本书部分程序进行了测试和验证工作，在此也一并向他们表示诚挚的感谢！

编　者

2019 年 7 月

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

# 目　录


第 1 章　大赛简介与经验交流 …… （1）
1.1　大赛简介 …… （1）
1.2　经验交流 …… （2）
1.2.1　意法半导体公司 …… （2）
1.2.2　宏晶科技公司（STC） …… （3）
1.2.3　北京国信长天科技有限公司 …… （4）
1.2.4　我对嵌入式人才的理解和期待 …… （5）
1.2.5　参赛经验交流 …… （5）
第 2 章　开发环境与工具 …… （8）
2.1　单片机设计与开发项目的开发环境与工具 …… （8）
2.1.1　Keil C51 集成开发环境 …… （8）
2.1.2　STC-ISP 程序下载软件 …… （12）
2.1.3　IAP15F2K61S2 程序调试方法 …… （14）
2.2　嵌入式设计与开发项目开发环境与工具 …… （18）
2.2.1　软件开发包（SDK） …… （18）
2.2.2　集成开发环境（IDE） …… （19）
2.2.3　Coocox 调试器设置 …… （21）
第 3 章　单片机设计与开发项目模块程序设计 …… （24）
3.1　LED 程序设计 …… （24）
3.2　中断程序设计 …… （26）
3.3　定时器程序设计 …… （29）
3.4　数码管程序设计 …… （34）
3.5　矩阵键盘程序设计 …… （38）
3.6　串口程序设计 …… （42）
3.7　DS1302 时钟芯片驱动程序设计 …… （45）
3.8　DS18B20 温度传感器驱动程序设计 …… （50）
3.9　PCF8591 ADC/DAC 驱动程序设计 …… （54）
3.10　AT24C02 存储器驱动程序设计 …… （57）
3.11　超声波传感器驱动程序设计 …… （60）
3.12　频率测量程序设计 …… （63）
第 4 章　单片机设计与开发项目往届试题解析 …… （64）
4.1　第七届预赛试题——模拟风扇控制系统 …… （64）
4.1.1　系统设计 …… （65）
4.1.2　系统实现 …… （69）
4.2　第七届决赛试题——电压频率采集设备 …… （69）
4.2.1　系统设计 …… （71）
4.2.2　系统实现 …… （78）
4.3　第八届预赛试题——电子钟 …… （78）


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

4.3.1 系统设计……（79）
4.3.2 系统实现……（84）
4.3.3 客观题……（84）
4.4 第八届决赛试题——超声波测距机……（85）
4.4.1 系统设计……（87）
4.4.2 系统实现……（91）
4.4.3 客观题……（91）
4.5 第九届预赛试题——彩灯控制器……（93）
4.5.1 系统设计……（94）
4.5.2 系统实现……（98）
4.5.3 客观题……（99）
4.6 第九届决赛试题——多功能测量仪表……（100）
4.6.1 系统设计……（102）
4.6.2 系统实现……（106）
4.6.3 客观题……（107）
4.7 第十届预赛试题……（108）
4.7.1 系统设计……（109）
4.7.2 系统实现……（113）
4.7.3 客观题……（113）
4.8 第十届决赛试题……（115）
4.8.1 系统设计……（118）
4.8.2 系统实现……（123）
4.8.3 客观题……（123）
**第 5 章 嵌入式设计与开发项目模块程序设计**……（126）
5.1 LED 指示灯程序设计……（126）
5.2 独立按键扫描程序设计……（128）
5.3 LCD 应用程序设计……（132）
5.4 USART 应用程序设计……（134）
5.5 AT24C02 存储器应用程序设计……（138）
5.6 ADC 应用程序设计……（141）
5.7 TIM 应用程序设计……（145）
5.8 数码管静态显示程序设计……（150）
5.9 ADC 键盘扫描程序设计……（152）
5.10 DHT11 湿度传感器程序设计……（156）
5.11 DS18B20 温度传感器程序设计……（158）
5.12 LIS302DL 加速度传感器程序设计……（160）
**第 6 章 嵌入式设计与开发项目往届试题解析**……（164）
6.1 第七届预赛试题——液位检测告警系统……（164）
6.1.1 系统设计……（165）
6.1.2 初始化程序设计……（167）
6.1.3 处理程序设计……（168）
6.1.4 系统实现……（173）



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

6.2 第七届决赛试题——温、湿度监控设备 …… （173）
6.2.1 系统设计 …… （175）
6.2.2 初始化程序设计 …… （178）
6.2.3 处理程序设计 …… （180）
6.2.4 系统实现 …… （185）
6.3 第八届预赛试题——模拟升降控制器 …… （186）
6.3.1 系统设计 …… （187）
6.3.2 初始化程序设计 …… （189）
6.3.3 处理程序设计 …… （190）
6.3.4 系统实现 …… （196）
6.3.5 客观题 …… （196）
6.4 第八届决赛试题——频率控制器 …… （198）
6.4.1 系统设计 …… （199）
6.4.2 初始化程序设计 …… （202）
6.4.3 处理程序设计 …… （205）
6.4.4 系统实现 …… （209）
6.4.5 客观题 …… （210）
6.5 第九届预赛试题——电子定时器 …… （211）
6.5.1 系统设计 …… （212）
6.5.2 初始化程序设计 …… （214）
6.5.3 处理程序设计 …… （215）
6.5.4 系统实现 …… （220）
6.5.5 客观题 …… （220）
6.6 第九届决赛试题——电子秤 …… （221）
6.6.1 系统设计 …… （223）
6.6.2 初始化程序设计 …… （225）
6.6.3 处理程序设计 …… （226）
6.6.4 系统实现 …… （232）
6.6.5 客观题 …… （233）
6.7 第十届预赛试题 …… （234）
6.7.1 系统设计 …… （235）
6.7.2 初始化程序设计 …… （237）
6.7.3 处理程序设计 …… （238）
6.7.4 系统实现 …… （242）
6.7.5 客观题 …… （242）
6.8 第十届决赛试题 …… （244）
6.8.1 系统设计 …… （246）
6.8.2 初始化程序设计 …… （248）
6.8.3 处理程序设计 …… （250）
6.8.4 系统实现 …… （256）
6.8.5 客观题 …… （256）

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

附录 A　国信长天单片机竞赛实训平台……（259）
附录 B　国信长天嵌入式竞赛实训平台……（264）
附录 C　嵌入式资源扩展板……（270）
附录 D　单片机设计与开发项目竞赛大纲……（277）
附录 E　嵌入式设计与开发项目竞赛大纲……（278）
参考文献……（280）



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

# 第 1 章　大赛简介与经验交流

本章包括大赛简介以及技术源厂商专家、用人单位专家和参赛获奖师生就单片机和嵌入式的学习及实践经验交流。

## 1.1　大 赛 简 介

为推动电子信息技术产业的发展，促进电子信息专业技术人才的培养，向电子信息技术行业输送具有创新能力和实践能力的人才，提升高校毕业生的就业竞争力，全面推动行业发展及人才培养进程，2009 年 1 月，工业和信息化部人才交流中心成功主办了首届全国电子专业人才设计与技能大赛，大赛先后被冠名为“天华杯”和“国信长天杯”。自 2013 年第五届大赛开始，全国电子专业人才设计与技能大赛和全国软件专业人才设计与创业大赛统一合并为全国软件和信息技术专业人才大赛，冠名为“蓝桥杯”。

一转眼，大赛走过了十年。作为大赛的初始参与者，我们非常欣喜地看到，在工信部和教育部相关领导的关心和指导下，在社会各界的支持和鼓励下，在参赛院校师生的信任和参与下，在大赛组委会同事们的努力和坚持下，大赛取得了极大的成功和发展，超过 1200 所高校的 30 万大学生先后参加了连续举办的十届大赛。

大赛的宗旨是“立足行业、突出实践、广泛参与、促进就业”，组委会以办教育的心态组织了首届大赛，十年来，也一直在忠实地践行着这些宗旨。

1）立足行业

大赛不是对现有教育体系的否定或简单地重复，而是对现有体系的有益补充。我们认为大学生培养的目标，是最终输出有益于行业和有益于社会的人才。那么大学的实践性教育，应该能与行业的技能需要进行良好的对接。大赛电子类科目服务的是电子信息产业，原则上产业对人才有什么需要，我们就按照科学规律考核什么，当然也因此希望学生学习、训练什么。

大赛组委会广泛咨询了高校师生、行业技术源厂商以及用人单位对电子类科目竞赛内容和形式的意见和建议。在此我们尤其感谢知名企业如 ARM 公司、意法半导体公司、宏晶科技公司和国信长天科技公司等，这些企业为大赛电子类科目的竞赛提供了非常多的建议和支持。

2）突出实践

大赛电子类科目关注的是实践型人才的培养。大赛的形式，是在封闭的环境中，5 个小时时间内，设计并调试成功一个具备一定商业价值的实际电子设计案例。大赛综合考核了电子工程师的理论知识、设计思路、调试经验以及熟练程度，尤其重视实践动手能力和工程系统思维能力。即便是博士或者硕士研究生参赛，如果平时没有做过科研项目或者缺乏相应的训练，也很难取得好的成绩。从另一个角度来说，能获得好成绩的参赛学生，将来有极大的概率成为一个优秀的工程师。

3）广泛参与

电子信息相关专业的学生，是一个规模庞大、层次丰富的群体。我们希望通过“蓝桥杯”大赛，对学生的专业学习起到激发兴趣、增强信心、树立目标和提供引导的作用。“蓝桥杯”大赛不是服务于少数精英学生的赛事，也不是特意为学校争取竞赛荣誉而设立的赛事，而是面向所有大学生且为提升他们的学习兴趣和专业技能而设立的赛事。在全面提升我们的智慧科技和现代制造业的今天，尤其需要从整体上提升我们信息技术从业人员的整体工程技术素质，培育和夯实科研

电子工业出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

和产业人才的基石。

大赛欢迎所有在校大学生参赛，也期待高校从培养人才的根本目标出发，鼓励和支持本校学生参赛。即便是条件还不够成熟，暂时不能参加大赛，我们也鼓励相关专业的学生们关注大赛，从大赛中主动获取学习资源，向大赛获奖选手学习，从而走上专业学习的快捷之路。

4）促进就业

检验教育成绩的一个重要标准是就业。真正做到了学以致用，将有益于学生，有益于用人单位，有益于产业的发展。十年大赛的一个重要成就，就是对参赛选手就业的显著促进作用；随着大赛的进一步发展，社会影响力持续发酵，这种促进作用还将进一步放大。

大赛对就业的促进作用，主要源自于以下因素：

（1）大赛“立足行业、突出实践、广泛参与”的指导精神，完美契合了“促进就业”的目标。

（2）大赛的竞赛内容和形式，有利于引导学生用正确的方法和形式，学习和训练合适的内容。绝大多数获奖学生都具备热爱电子信息行业、基础扎实、实践动手能力强的特征；即便是没有获奖的学生，在经历了长达一年的学习准备后，也收获了学习的自信，明白了差距，清楚了自己未来努力和发展的方向。

（3）大赛以个人为单位全封闭竞赛，强调和体现了公平性、公正性和客观性，使得获奖选手在用人单位获得了良好的口碑，继而促进了选手的就业，使大赛步入良性循环。

深刻地理解这些大赛的宗旨，有助于电子类参赛师生利用这本大赛指导书准备竞赛，也有助于学生专业素质和技能的提高。

“蓝桥杯”大赛也是一个良好的交流平台。大赛并没有直接培养如此多的人才或者直接提升了他们的就业能力，成绩主要是通过参赛选手自己的不懈努力和指导教师的辛勤培育获得的，但让我们感到欣慰的是，通过大赛的筛选机制，我们为这些成绩添砖加瓦，营造了一个更加丰满的电子信息技术类人才培养生态。这也是“蓝桥杯”大赛的本意，大赛已经成为连接信息技术行业主管单位、行业协会、教育机构、技术源厂商、用人单位和人才及高校的桥梁。

我们关注的不仅仅是参赛选手，我们希望更多的同学来关心“蓝桥杯”大赛，主动获取更多的学习资源。对“蓝桥杯”大赛感兴趣的师生，以及对“蓝桥杯”大赛人才培养理念认同的师生，可以参考本书，结合大赛组委会官方网站“www.lanqiao.org” 以及国信长天公司的官方网站“www.gxct.net”，获得相应的学习、训练及参赛的信息和资源。

“蓝桥杯”大赛是一个务实的大赛，也是一个开放的大赛和与时俱进的大赛，在大赛即将进入第十一届新的竞赛周期的时刻，我们将认真总结过往十届大赛的经验和教训，以电子信息技术行业实践型人才培养的客观规律和现实需求为出发点，将大赛现有的单片机和嵌入式两个基础的电子类科目办好；同时我们也将密切关注行业发展趋势和新的技术成果，探索对物联网和智能制造等应用型科目进行专业评测和竞赛的可行性。

## 1.2 经 验 交 流

本节包括技术源厂商专家、用人单位专家和参赛获奖师生等就单片机和嵌入式的学习及实践经验的交流。

### 1.2.1 意法半导体公司

意法半导体公司中国区微控制器事业部市场及应用总监 曹锦东

世界万物，智能互联，打造数字化、智能化的世界是行业发展的方向和趋势。

技术的升级和工艺的进步，使得数字化的基础器件，比如传感器和微控制器在性能、封装和



电子工业出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

成本上越来越能够满足产业严苛的要求；这些进步将加速万物互联的进度，万物互联后产生的大数据可以进一步提升社会效率并推动产业升级，将产生巨大的社会价值。

产业升级，技术创新，方案落地，离不开与时俱进的人才。高等学校是人才最大的培养基地。作为致力于长期服务中国市场、为中国的产业发展提供最新技品的合作伙伴，意法半导体一直为中国的用户提供最前沿的技术、推动生态系统的建设，为用户提供从芯片到应用的支持。

为帮助产业界培养人才，意法半导体从 2011 年开始联合工信部人才交流中心举办全国性的嵌入式设计科目竞赛（“蓝桥杯”大赛），通过竞赛平台，让学生掌握前沿技术，培养能够适应社会岗位的应用型人才。

“蓝桥杯”大赛是意法半导体公司中国大学计划的重要组成部分，这个竞赛非常好地提升了学生的动手能力，同时也促进了学生学习前沿技术的积极性。

为全方位推动人才培养，意法半导体公司将依托工业和信息化部人才交流中心、“蓝桥杯”大赛等平台开展以下工作：

（1）推动精品课程建设、协助高校课程改革，将最前沿的技术和产品带入教学和实验中，让学生接触到、体验到最新技术，为以后就业打好基础。

（2）实施 TTT 项目（老师培训老师），邀请有开课经验的老师开展培训、帮助打算开课的老师提升信心，分享教育经验和体会。

（3）开展大学生创新创业实践项目，让学生们所学习和掌握的知识和能力，通过大赛进一步夯实，在一个公平的环境里，模拟企业项目，提升自身能力和信心。

“蓝桥杯”大赛以其独特务实的竞赛形式和严谨公正的竞赛组织，对中国实践型电子信息技术人才的培养，起到了非常积极的促进作用。大赛参赛人数逐年增加，同时通过竞赛锻炼的学生，多数就业前景良好。借此机会，我们非常感谢工信部人才交流中心，十年如一日，以服务产业为目标，为扩大竞赛的影响力和参与度而不懈努力。

展望未来，产业需要更多人才，我们将联合工业和信息化部人才交流中心和大赛组委会，全面启动人才培养计划，协助高校升级课程体系，鼓励学生积极参加各项竞赛，以赛代练，提高技术能力，为产业更好地服务。

### 1.2.2 宏晶科技公司（STC）

宏晶科技创始人　姚永平

STC 自成立起就一直专注于国内流行的 8051 单片机的研发工作，和各高校以及企业保持着密切的联系和紧密的合作。根据 STC 与高校及企业合作多年的经验，我们深刻感受到当前大学里单片机课程在开设的时间、顺序以及完整性方面存在弊端，特提出以下教学改革建议：

首先，我们建议“C 语言程序设计”课程应在大学一年级第一学期开设，很多高校已这样做了，但还有部分学校滞后了。我的观点是晚学不如早学，工科、非工科都应将此门课作为进入信息时代的计算机扫盲课，学好了“C 语言程序设计”课程就打好了单片机学习中的部分软件基础。相对于用汇编语言开发程序，C 语言将程序开发难度降低了很多，对开发人员来说，效率得到极大的提高。

其次，我们建议将“数字电路”课程放在大学一年级第一学期开设，“数字电路”能帮助学生理解微机原理和基本的数字系统，并能设计一些简单的硬件电路系统，是单片机学习的硬件基础。

接下来就要学习“微机原理及单片机应用”了，我们建议将“微机原理及单片机应用”课程提前到大学一年级第二学期开设。许多学校的相关工科专业都是在大学三年级第二学期才开设这门课程。由于接触单片机的时间晚，学生没时间认真学习和研究单片机，更没时间应用所学的知识开发一些单片机项目。许多学生在只学到一些单片机皮毛的情况下，便匆匆步入社会进入了相

电子工业出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

关行业。这种情况往往导致很多学生不能胜任自己的岗位，工作起来很吃力，而且极容易被淘汰。这就有了“学生抱怨就业难，企业却反映招不到人才”的怪现象。

如果将“微机原理及单片机应用”课程提前至大学一年级第二学期开设，学生及早接触单片机，那么对单片机有兴趣或以后想从事相关行业的同学在意识到单片机的重要性后，就有充足的时间去深入研究单片机，从而将单片机系统设计搞熟。这样在他们毕业以后进入相关行业就能很快胜任工作。

单片机课程不是一门纯理论性的课程，还需要学生去动手实践和开发，所以学习单片机以及开发单片机项目能提高学生的创新能力，如果学得太晚就很难学以致用。其实很多工科的专业课程都是如此，很多学生在学习完这些专业课程后之所以不知所云，就是因为没有花时间或没有足够的时间去研究和实践所学的内容。如果这些实践型的专业课程开设得太晚，学生在学习完之后根本没有时间去动手应用的话，那么这些课程就形同虚设，对学生以后的就业将起不到应有的帮助。

另外，为了使学生在开发程序时能得心应手，我们建议在大学一年级第二学期应开设“数据结构”课程，以提高开发效率，降低开发难度。如果“C 语言程序设计”没学好，通过对“数据结构”的学习，也可算对“C 语言程序设计”的复习，为单片机开发打好软件基础。

最后，我们建议在大学二年级第一学期开设“实时操作系统”和“模拟电路”等后续课程。这些课程是对前面所学课程的补充和提高，如果“数据结构”没学好，学习“实时操作系统”也可算对“数据结构”的复习。基于实时操作系统的单片机开发，对开发者的能力要求可大幅降低，能较容易地开发出大型、复杂的单片机项目。

乘风破浪会有时，直挂云帆济沧海。相信单片机课程教育的改革会给当前的单片机教育和科技创新开创一片新的天地，相信各高校一定能培养出更多、更专业、更符合社会需求的优秀人才。

### 1.2.3 北京国信长天科技有限公司

北京国信长天科技有限公司　乔婷

北京国信长天科技有限公司是专业的教育科技服务企业，聚焦于物联网、智能制造等电子信息应用技术领域。公司伴随着“蓝桥杯”大赛而成立，一直在以下三个方向默默耕耘。

（1）承接和支撑了大量的工业和信息化部人才交流中心的教育、培训和人才服务项目。

该公司是“蓝桥杯”全国软件和信息技术专业人才大赛技术支持单位。其中 2011 年第三届及 2012 年第四届大赛即被冠名为“国信长天杯”大赛。

该公司是工业和信息化部人才交流中心嵌入式学院项目承办单位。

该公司是工业和信息化部、人力资源与社会保障部信息技术人才知识更新工程“全国电子专业人才考试”项目技术支持单位。

该公司是工业和信息化部人才交流中心“全国电子专业人才师资培训班”项目承办单位。

（2）依托“蓝桥杯”大赛平台，与众多国内外著名的信息技术企业和研究机构，如西门子公司、ARM 公司、意法半导体公司、Altium 公司和宏晶科技公司等，建立了广泛的合作关系，为以上公司的技术、产品、实验室以及大学计划在高校中的推广，做出了重要的贡献。

（3）设立了专业的研究机构，为物联网、智能制造等电子信息行业应用型人才的培养研发并建设了相应的专业建设体系（含课程、师资、实验实训设备等）和人才评测体系。

对于电子类专业人才的教育培养，北京国信长天科技有限公司认为要重视基础性专业知识的教育和动手能力的培养。一个电子信息类技术人才，无论是从事系统设计，还是从事产品功能模块的开发，或者从事系统的安装调试、运行维护、维修保障等，都需要掌握包括电子电工、单片机、C 语言、电路板设计与焊接调试、基本的工具软件和测试仪表的使用等基础知识和技能。掌



电子工业出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

握这些知识和技能不难，但需持之以恒地坚持。以我们十多年“蓝桥杯”大赛服务的经验，达到熟练掌握程度的学生不多，甚至一些授课教师都远远达不到深刻理解和熟练掌握的程度。究其原因，是电子信息类知识和技能种类丰富，有很强的系统性和关联性，需要长时间积累，对于爱好这个专业的人来说，这种积累过程是轻松甚至是令人愉悦的；而对于一般人来说，这个过程是枯燥和艰难的。所以我们对电子类专业人才的培养，首要的是培养学习的兴趣，把学生引导到专业学习的路上来。“蓝桥杯”大赛倡导的就是这种理念和方法。

北京国信长天科技有限公司在做好“蓝桥杯”相关服务的同时，也希望在培养学生学习兴趣，指导学生走好嵌入式、物联网、智能制造等电子类专业人才学习成长的第一步这个领域添砖加瓦。伴随着“蓝桥杯”大赛的成长，我们专心做了十多年的电子信息技术人才的评测工作。下一步，本公司将继续加大投入、广纳英才，研究电子类专业技术人才评测的标准，在嵌入式、物联网、智能制造等领域持续建设专业的人才技能评测知识库，为中国电子类专业人才的培养贡献我们的微薄之力。

### 1.2.4 我对嵌入式人才的理解和期待

中国航天科工集团二院 23 所专业副总师　李冬伍

随着物联网概念的兴起与广泛应用，传感器、控制技术、云计算、大数据和移动互联等基础应用技术的不断发展，以轻智能与智能硬件单品厂商为代表的创新创业团队层出不穷，不断改变人们过去对嵌入式系统应用的认识。行业的不断发展带动了企业对嵌入式人才的需求，这对在校学生来说是件好事，但从用人单位的角度来看，对嵌入式人才的能力要求也越来越高，毕业时能否满足企业的用人要求，是求职者是否拥有竞争力的关键要素。

嵌入式系统是一种量身定制的系统，行业对人才的需求可以简单概括为系统集成设计人员、驱动程序开发人员、应用程序开发人员三类，工作过程中需要面对不同的硬件环境和不同的配置需求，要求开发人员具有一定的实战经验和良好的动手能力、学习能力。企业招聘过程中也正在逐渐加强对应聘者动手能力的考核，要求应聘者在真实的环境下进行软件编程和硬件调试。当前，尽管众多高校的电子、计算机专业开设了程序设计、微机原理等课程，但高校输送的人才与企业需要的人才没有很好地对应上，学与用之间存在一定程度的脱节。

在人才招聘的实践过程中，我很偶然地接触到“蓝桥杯”大赛电子类竞赛的获奖选手并由此了解了“蓝桥杯”大赛，感受到“蓝桥杯”大赛是非常好的练习和提升实操能力的平台，希望有更多的同学积极地参与进来，利用这些平台，将所学理论知识运用到实践中，提升自己的软硬件设计和调试能力。不管是研究型高校还是应用型高校的毕业生，将来无论是从事综合性系统设计工作，还是应用型技术服务工作，熟练掌握专业基础知识且拥有出色的动手能力，都将有助于在本职工作中脱颖而出。对于我个人来说，熟练掌握至少一门主流的 32 位微处理器的使用，并拥有数万行嵌入式代码编写经验的人，才有可能被称之为嵌入式人才。

### 1.2.5 参赛经验交流

成都信息工程大学电子实验中心　杨明欣

“蓝桥杯”大赛已成为高校广泛参与、行业认可度高、影响力不断扩大的行业竞赛。竞赛推动了全国高校信息与电子类学科课程体系和课程内容的改革，有助于高等学校实施素质教育，培养大学生的创新能力、协作精神和理论联系实际的学风；有助于学生工程实践素质的培养、提高学生针对实际问题进行电子设计制作的能力；有助于吸引、鼓励广大青年学生踊跃参加课外科技活动，为优秀人才的脱颖而出创造条件。

成都信息工程大学自 2010 年（第二届）开始，连续参加一年一度的“蓝桥杯”大赛并取得了

电子工业出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

优异的成绩和良好的育人效果。以下为我们对近年来参加“蓝桥杯”大赛的简要总结。

1）竞赛组织

我校竞赛组织实施工作由教务处与校团委牵头，电子实验中心在相关学院及电子协会的支持下具体组织实施，通过多年的运作与改进，形成了比较完善的学生课外科技活动体系。在学校政策及制度保障下，电子实验中心依托电子协会，以“蓝桥杯”大赛等学生科技活动为目标牵引，积极开展以课外学术科技创新为主题的校园社团活动，对学生的工程能力培养及创新教育发挥重要的导向作用。

在日常训练活动安排上，采用了由电子实验中心进行指导并提供场地设备、教务处在经费上进行支持、以校电子协会为主体的组织形式，举办由中心教师及协会骨干学生主讲的系列专题讲座，并开展器件代购和设计指导等服务，为学生提供方便。在此基础上定期组织专题制作比赛及家电维修和三下乡等活动，以各种形式锻炼学生的能力。对创意好的学生自拟项目，学院在技术指导、场地和器件上给予支持，培育参与挑战杯活动。

2）参赛收获

参加“蓝桥杯”大赛，不仅使我们收获了优异的竞赛成绩，还从整体上促进了我校学生的成才和师资队伍的成长。

（1）大赛推动了学生的科创活动及就业能力。

学生课外科技创新实践活动在经费保障、创新实验室条件保障、活动安排等方面推行与社会接轨方式，电子协会通过走出校门，走向社会，深入社区来了解社会；也通过参加各种有挑战性的竞赛活动提高了学生的创新能力。大赛促进了学生独立思考，大胆探索，积极提出自己的新思想、新观点、新思路、新设计的热情；激发了学生的好奇心和求知欲，促进了学生探索精神和创新思维的形成。

电子协会的学生通过系列学习，技术能力及综合素质得到明显提高，也使得学生有更多机会与教师交流，教师在科研中吸收能力强的学生参与课题研究，近年来电子信息类专业近 10%的大四学生参与教师的助研活动并发表了多篇具有实用价值的文章。

同时学校采用灵活机制鼓励学生个人创业，已经有部分学生组成了创业小组。学生在校期间进行自主创业的尝试，成功推出了系列电子产品，将所学知识转化为社会生产力。鼓励学生参与教师科研、承接社会课题等方式，缩短了学生在校学习与将来社会就业的过渡距离。据不完全统计，电子协会创新实验室的骨干学生，毕业三年以后自主（或合伙）创业的学生比例达 15%以上。

“蓝桥杯”大赛活动的开展，使我院电子类专业学生的创新精神和实践能力普遍提高，综合素质不断增强，受到用人单位的高度评价和社会各界的充分肯定，推动了学生就业及自主创业活动。参加课外科技活动的学生，深受用人单位的欢迎。近五年，参加过竞赛的毕业生就业情况良好，就业单位包括大疆、天瑞仪器、广州周立功公司、利尔达、29 所、总参 57 所等知名单位。

同时，每年几百个参加竞赛的学生在职场的良好表现，提升了我校的社会声誉及我校学生的整体就业竞争力，形成了良好的品牌效应。同时竞赛受益学生在实现自我价值的过程中，也抱着感恩之心积极宣传学校，为学弟学妹提供机会，回报学校，拓展了我校学生的就业渠道。

（2）大赛促进了师资队伍的建设。

高校工程教育的关键在于具有工程经验教师队伍的建设。“蓝桥杯”大赛在运行过程中，学生的热情很高，但由于学生流动性大，缺乏相应的经验积累，要保证“蓝桥杯”大赛健康有序发展，必须精心挑选培养协会的辅导教师团队。为此，学校为鼓励教师参与课外科技活动，在职称评定及年度考评上给予政策倾斜，对指导成绩优秀的教师在经济上给予奖励；同时多年来教师团队积极引进工程经验丰富的教师，邀请校外知名人士定期到校讲座指导；并要求青年教师必须参加指导教师团队；在队伍组成上新老搭配，依据教师的专长、学科方向和兴趣，分别承担相应分会的



电子工业出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

指导任务，促进了青年教师成长。

竞赛活动的开展，拉近了师生的距离，形成了“亦师亦友”的新型师生关系，教师更深入地了解了教学中存在的不足，推动了教学改革，有力促进了“爱生、敬业、求实、创新”教风的形成。

“蓝桥杯”大赛指导教师积极参与学生指导工作，达到了因材施教的目的。指导教师积极进行教改活动，申报教改课题，主持多项省级、校级教改项目。教改项目的进行既锻炼了教师队伍，也为提高教学质量、改善实验室环境创造了条件，有效地解决了课程建设乃至本科教育过程中不断遇到的新问题。教学队伍在实施研究与改革的同时，及时将成功的经验付诸文字，发表在有影响的期刊上。近 5 年来发表教改教研论文多篇，也在与同行的交流中进一步找到差距，使课程的改革与建设不断完善，同时编写了教材及实验指导书。

竞赛要求指导教师要有很宽的知识面和丰富的实践经验。教师在指导学生的过程中相互学习，共同提高，知识和经验的相互渗透和补充使得指导教师的业务素质有了很大的提高，这无疑使指导教师能够在今后的日常教学特别是实践教学和科研中发挥更重要的作用。

同时竞赛培训活动的开展，也作为学生实习实训的有效载体，较好地完成了工程教育中要求加强学生工程实训的要求，一定程度上解决了当前教学过程中实习公司落实难度大、效果不佳的问题。

（3）大赛促进了学风建设。

“蓝桥杯”大赛的开展，巩固提高了第一课堂教学效果，促进了第一课堂的教改；形成了与第一课堂良性互动的局面。“蓝桥杯”大赛的良性发展，离不开学校的管理及社会对参与师生的有效认可。由于竞赛目标明确，社会认可度高，每年有规律的时间安排特别适合学生的培养。一年左右的完整参与周期，可以使参与该活动的学生直接看到成效，充分调动了学生的学习兴趣及潜能，其动手能力、创新能力、团队协作能力、知识综合应用能力等多方面都能得到锻炼，对提高学生的工程素质及创新能力，取得了课程教学难以达到的良好效果。

同时受传统教育模式的影响，学校的课程设置偏重传统学科、传统技术及理论教学，忽视了学科之间的交叉整合及实践教学；缺乏弹性的教学安排使得批量化实验室建设难以适应差异化的学生创新实践要求；“蓝桥杯”大赛灵活多样化的创新实践平台，促进了实践教学改革及重视实践创新的良好学风。

“蓝桥杯”大赛促进了学生独立思考、大胆探索的热情，激发了学生的好奇心和求知欲，促进了学生探索精神和创新思维的形成。

电子协会全天候开放实验室，为学生的创新活动提供了良好的场所，学生可根据自己的时间安排进入实验室学习，不同学科、不同年级、不同个性的学生在这里自由交流，以独立或合作方式参与项目开发，构造一个和谐、开放的学习氛围。

学校将课外科技活动、社会实践、学生社团活动纳入学分计算；同时电子协会在进行各项活动开展的同时，把学生“蓝桥杯”大赛培训作为主要抓手，由于参赛学生就业情况好，使得其他学生参与协会活动的热情高涨。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

# 第 2 章　开发环境与工具

本章介绍单片机设计与开发项目和嵌入式设计与开发项目的开发环境与工具。

## 2.1　单片机设计与开发项目的开发环境与工具

单片机设计与开发项目开发环境与工具包括 Keil C51 集成开发环境、STC-ISP 程序下载软件和国信长天单片机竞赛实训平台（参见附录 A）等。

### 2.1.1　Keil C51 集成开发环境

Keil C51 是美国 Keil Software 公司出品的 51 系列兼容单片机 C 语言软件开发系统，提供了包括 C 编译器、宏汇编、链接器、库管理和一个功能强大的仿真调试器在内的完整开发方案，通过集成开发环境（μVision）将这些部分组合在一起。Keil 兼容 C 语言和汇编语言，与汇编语言相比，C 语言在功能、结构性、可读性、可维护性上有明显的优势，因而易学易用。

开发人员可用 IDE 本身或其他编辑器编辑 C 语言（C51）或汇编语言（A51）源文件。然后分别由 C51 及 A51 编译器编译生成目标文件（.obj）。目标文件可由 LIB51 创建生成库文件，也可以与库文件一起经 L51 连接定位生成绝对目标文件（.abs）。abs 文件由 OH51 转换成标准的 hex 文件，以供调试器 dScope51 或 tScope51 进行源代码级调试，也可由仿真器直接对目标板进行调试，还可以直接写入程序存储器如 EPROM 中。

Keil μVision5 集成开发环境 V9.56 的安装文件是“Keil.C51. V9.56.exe”，双击该文件启动安装，将 Keil C51 安装在“C:\Keil_v5”文件夹中。

Keil μVision5 集成开发环境的使用方法是：在 Windows 操作系统“开始”菜单下的“所有程序”中找到“Keil uVision5”程序，或单击桌面上的“Keil uVision5”图标，运行“Keil uVision5”，进入 Keil μVision5 集成开发环境主界面。

下面以“3.1 LED 程序设计”为例，介绍如何使用 Keil μVision5 集成开发环境建立工程。

单击“Project”菜单下的“New μVision Project…”菜单项启动新工程的建立，出现新建工程对话框。为了便于工程管理，对于每个工程可以新建一个文件夹，例如本例中新建文件夹“D:\CT107D\301_LED”，进入“301_LED”文件夹后在文件名文本框中输入工程名称（例如“IAP15”），如图 2.1 所示。

单击“保存(S)”按钮，出现选择器件对话框，选择“Atmel”公司的“AT89C52”单片机，如图 2.2 所示。单击“OK”按钮，提示是否加载启动代码，如图 2.3 所示。单击“否(N)”按钮，出现如图 2.4 所示主界面。界面左侧的工程工作区出现“Target 1”文件夹，单击“Target 1”之前的“+”展开“Target 1”，出现下一级文件夹“Source Group 1”。

单击“File”菜单下的“New...”菜单项或单击文件工具栏中的“New”按钮，新建文件“Text1”，在“Text1”中输入“3.1 LED 程序设计”中的程序，单击“File”菜单下的“Save”菜单项或单击文件工具栏中的“Save”按钮，打开另存为对话框，如图 2.5 所示。

在另存为对话框的文件名中输入文件名“main.c”，单击“保存(S)”按钮保存文件。也可以先保存文件，再编写程序，这样可以使编写程序中的关键字或常数等以特殊颜色显示。



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

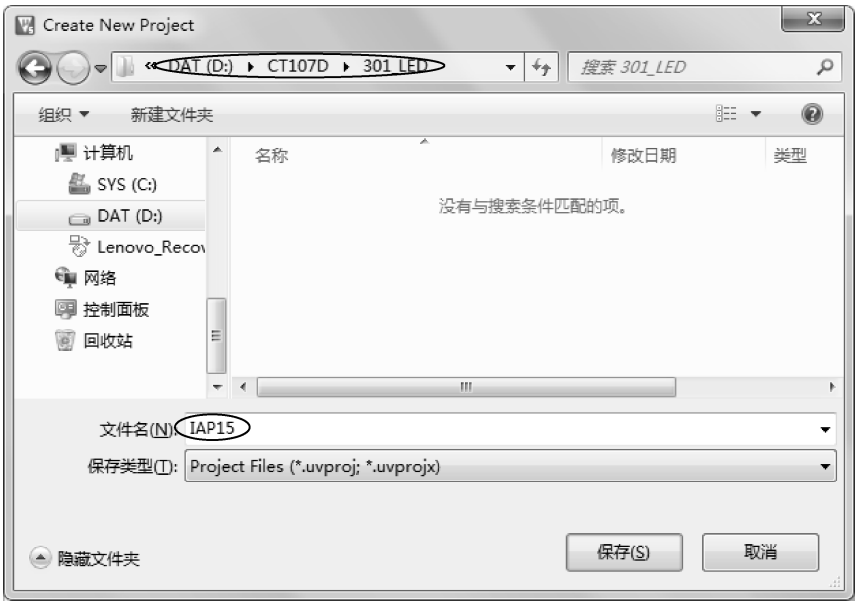



图 2.1　新建工程对话框

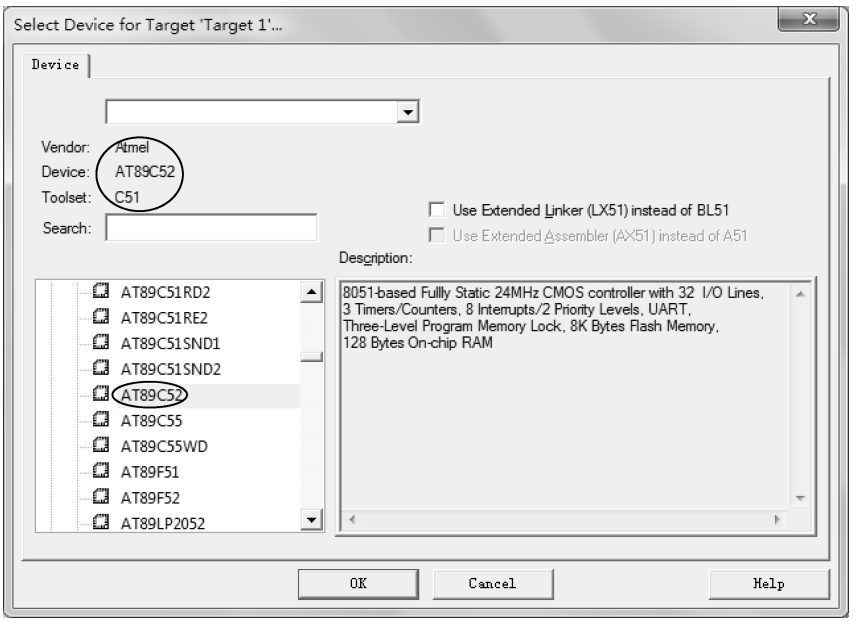



图 2.2　选择器件对话框

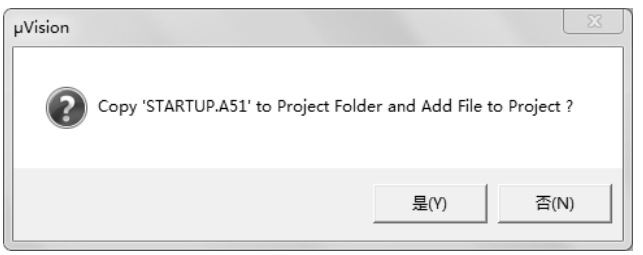



图 2.3　加载启动代码提示

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

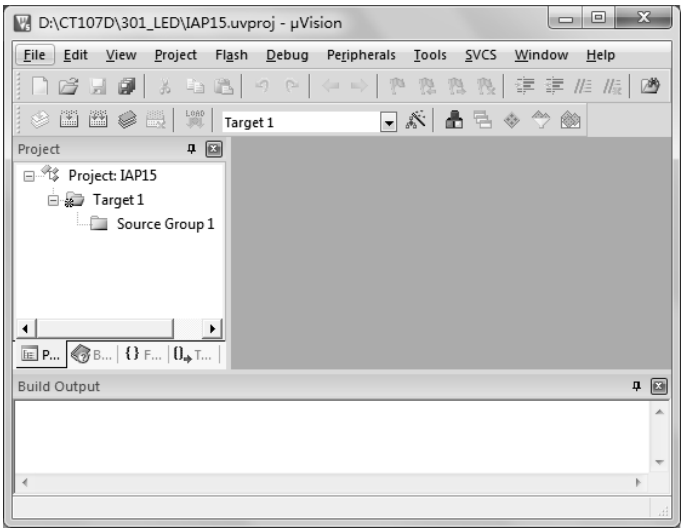



图 2.4　建立工程后的集成开发环境主界面

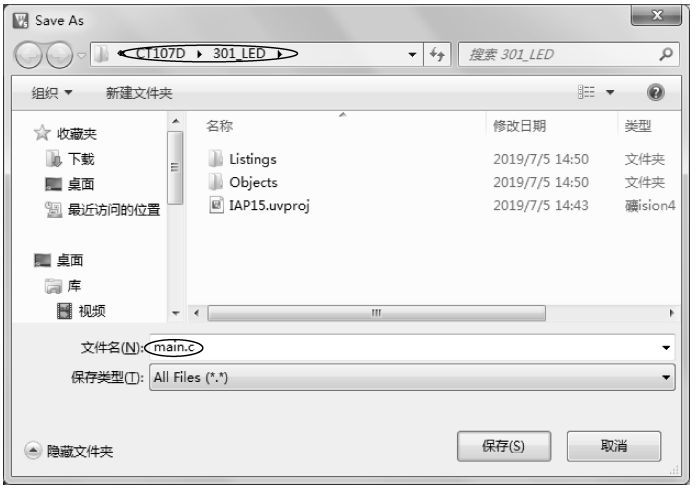



图 2.5　另存为对话框

Keil uVsion5 集成开发环境支持 C51 和汇编语言，如果使用 C51 语言编程，保存文件扩展名为“.c”；如果使用汇编语言编程，保存文件扩展名为“.asm”。

保存文件后，还需要将该文件添加到工程中，方法是：右击“Source Group 1”文件夹，在弹出的菜单中单击“Add Existing Files to Group 'Source Group 1'”，出现如图 2.6 所示的添加文件到组对话框。

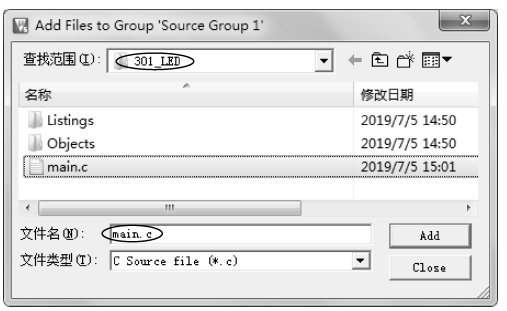


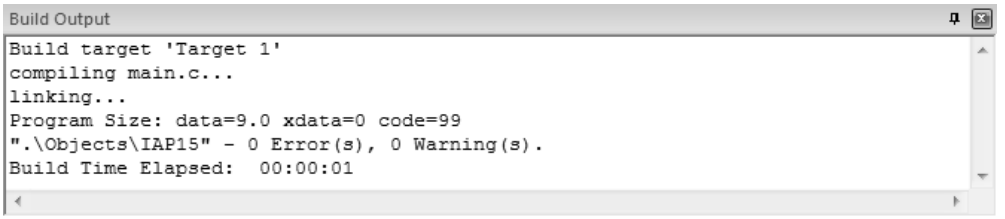

图 2.6　添加文件到组对话框



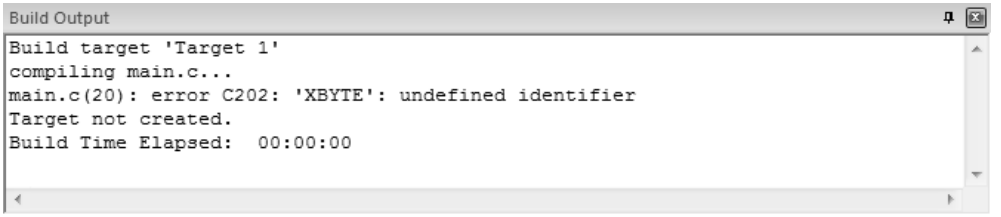
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

选择其中的“main.c”文件，单击“Add”按钮将“main.c”文件添加到工程中，单击“Close”按钮关闭添加文件到组对话框。

单击“Project”菜单下的“Build Target”菜单项或单击生成工具栏中的“Build”按钮编译工程，主界面下方的输出窗口显示编译结果。如果编译正确，可以看到提示 0 个错误与 0 个警告，如图 2.7（a）所示；如果源程序中有语法错误，会在主界面下方的输出窗口中提示发生错误或者警告，如图 2.7（b）所示，双击某一行，根据错误提示信息查找错误并纠正错误后重新编译，直到编译正确为止。




（a）编译正确




（b）编译错误

图 2.7　编译结果

编译正确后，单击“Debug”菜单下的“Start/Stop Debug Session”菜单项或单击文件工具栏中“Start/Stop Debug Session”按钮进入调试界面，如图 2.8 所示。

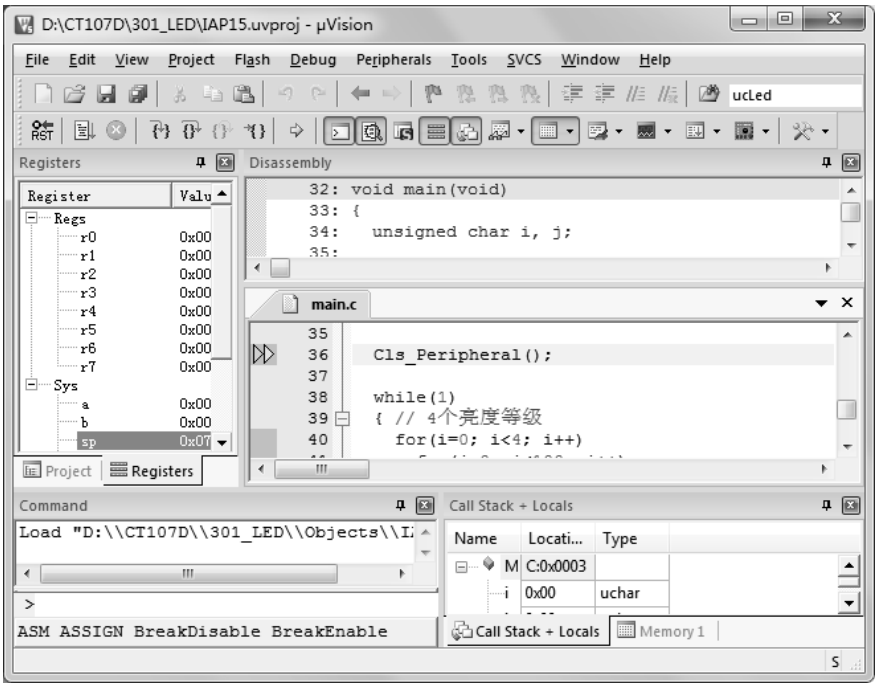



图 2.8　调试界面

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

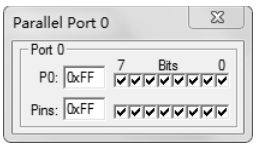



图 2.9　并口 0 对话框

调试界面的左侧显示相关寄存器的内容，比如 r0～r7、a、dptr、pc 和 psw 等，可以通过观察这些寄存器内容的变化判断程序功能的正确性。

单击“Peripherals”菜单下“IO-Ports”中的“Port 0”菜单项，打开并口 0 对话框，如图 2.9 所示。

单击“Debug”菜单下的“Run”菜单项或单击调试工具栏中的“Run”按钮运行程序，可以看到对话框中 P0 的值在全 1 和全 0 间不断变化。

**注意：**使用 MM 模式时 P0 的值一直为 0。

## 2.1.2　STC-ISP 程序下载软件

以下再以“3.1 LED 程序设计”为例，介绍如何使用 STC-ISP 程序下载软件向开发板下载程序，并进行硬件调试配置。

**1）目标选项设置**

默认情况下，Keil 项目编译后不会产生目标代码 HEX 文件，为了使项目编译后自动生成 HEX 文件，需要进行目标选项设置。

右击左侧窗口中的“Target 1”，在弹出的菜单中单击“Option for Target 'Target 1'”，或者单击“Project”菜单下的“Option for Target 'Target 1'”菜单项，或者单击生成工具栏中的“Option for Target...”按钮，打开目标选项对话框，在目标选项对话框中单击“Output”标签，选择“Create HEX File”选项，如图 2.10 所示。

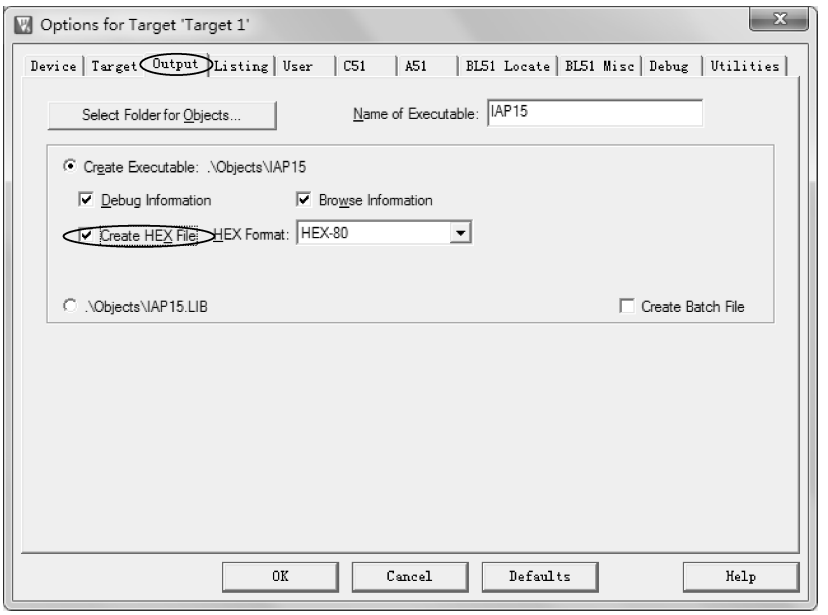



图 2.10　输出选项设置

设置完成后，重新编译工程，在工程所在文件夹的“Objects”文件夹中可以找到生成的 HEX 文件，文件名和工程名相同。

**2）USB 转接串口号识别**

电脑和单片机竞赛实训平台的通信通过 USB 转接串口实现，将单片机竞赛实训平台通过 USB 与电脑相连（注意先不要给开发板通电，即电源指示灯不应点亮。如果电源指示灯点亮，则按一下电源开关，将电源供电关闭），Windows 会自动安装驱动程序，如果不能自动安装，可以执行 CH341SER.exe 手动安装。



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

在“计算机管理”的“系统工具”中单击“设备管理器”，打开设备管理器，单击“端口（COM 和 LPT）”前的“+”号，可以查看 USB 转接的串口号，如图 2.11 所示。记住这个串口号（COM3：实际串口号可能不同），在接下来的 STC-ISP 软件中，设置的串口号必须和这里查看的串口号一致（实际上 STC-ISP 可以自动识别这个串口号）。

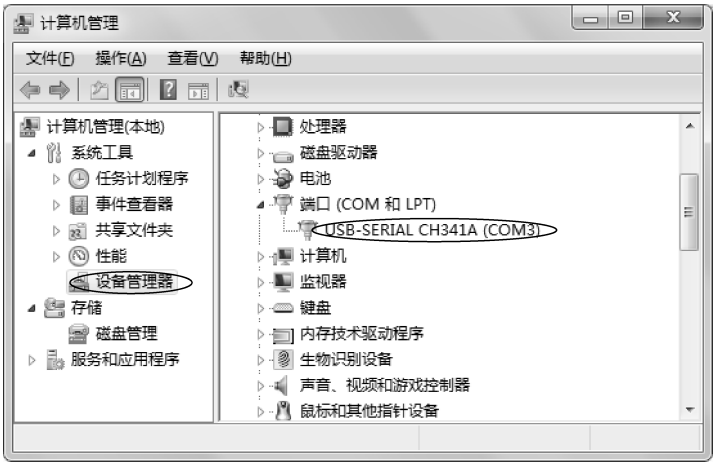



图 2.11　查看串口号

### 3）STC-ISP 程序下载

STC-ISP 的执行文件是 stc-isp-15xx-v6.86.exe，程序下载步骤如下。

（1）打开下载界面：双击 stc-isp-15xx-v6.86.exe 运行程序，出现如图 2.12 所示界面。

图 2.12　STC-ISP 界面

（2）选择单片机型号：在左上方的“单片机型号”下拉列表中选择单片机型号，单片机竞赛实训平台上的单片机型号为“IAP15F2K61S2”。

（3）确认串口号与计算机系统识别一致：将开发板通过 USB 与电脑相连，在“串口号”下拉列表中选择“USB-SERIAL CH341A (COM3)”。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

（4）选择程序文件 HEX 代码：单击“打开程序文件”按钮，打开工程文件夹“D:\CT107D\302_LED”下“Objects”文件夹中的“IAP15.hex”文件，界面右上方的程序文件标签中出现加载的 16 进制程序代码。

（5）选择 IRC 频率：在“硬件选项”中选择 IRC 频率为“12.000”MHz。

（6）下载程序代码：确认开发板未通电，即电源灯未点亮，单击界面左下方的“下载/编程”按钮，界面右下方显示“正在检测目标单片机”，如图 2.13 所示。

图 2.13　单击“下载/编程”按钮后的提示信息

按下开发板上的电源开关，开发板电源指示灯点亮，STC-ISP 检测到单片机，显示当前芯片的硬件选项，并开始下载程序，下载完成后显示“操作成功”，如图 2.14 所示。同时单片机竞赛实训平台运行程序，LD1～LD8 以不同亮度显示。

图 2.14　程序下载成功后的提示信息

如果操作不成功，请检查是否正确安装了驱动程序，可以打开设备管理器查看是否正确分配了串口号，且 STC-ISP 软件中设置的串口号是否和设备管理器查看到的串口号一致。

### 2.1.3　IAP15F2K61S2 程序调试方法

程序调试是程序设计的重要步骤，通过调试，不仅可以验证程序的功能，更重要的是发现和纠正程序中的功能错误。

**1）安装 Keil 版本的 STC 仿真驱动**

在 STC-ISP 右上方选择“Keil 仿真设置”标签，单击“添加型号和头文件到 Keil 中”按钮，打开浏览文件夹对话框，在对话框中选择 Keil 安装文件夹“C:\Keil_v5”，单击“确定”按钮，STC-ISP 提示“STC MCU 型号添加成功!”，如图 2.15 所示。

STC MCU 添加成功后，“C:\Keil_v5\C51\BIN”文件夹中出现 STC Monitor51 仿真驱动程序“stcmon51.dll”，同时“C:\Keil_v5\C51\INC\STC 文件夹中出现 STC 头文件，在 Keil 中新建工程选择芯片型号时，便会有“STC MCU Database”选项，从 MCU 列表中选择 MCU 型号（目前 STC 支持仿真的型号只有 STC15F2K60S2），如图 2.16 所示。



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

图 2.15　Keil 仿真设置界面

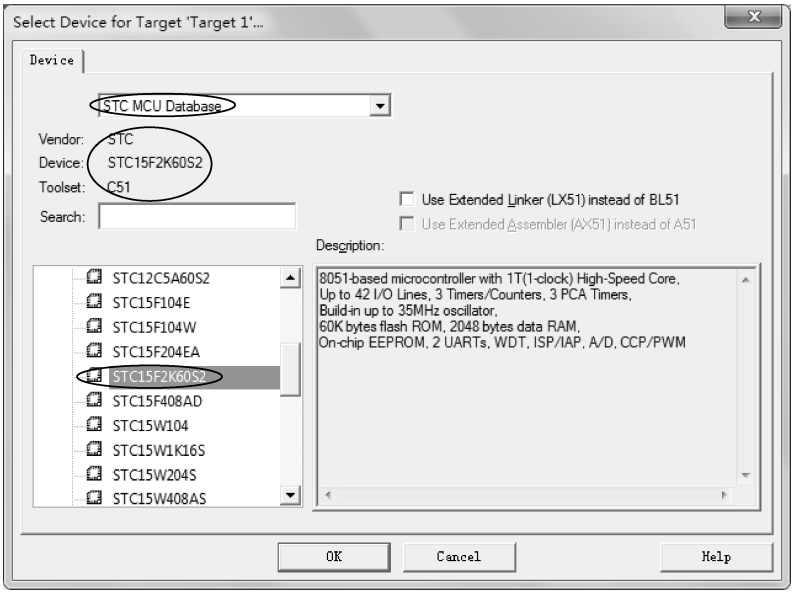



图 2.16　选择器件对话框

**注意**：当创建的是 C 语言工程，且将启动文件“STARTUP.A51”添加到工程中时，里面有一个命名为“IDATALEN”的宏定义，如图 2.17 所示，它是用来定义 IDATA 大小的一个宏，默认值是 128，即十六进制的 80H，同时它也是启动文件中需要初始化为 0 的 IDATA 的大小。所以若 IDATA 定义为 80H，STARTUP.A51 里面的代码则会将 IDATA 的 00-7F 的 RAM 初始化为 0；同样若将 IDATA 定义为 0FFH，则会将 IDATA 的 00-FF 的 RAM 初始化为 0。

虽然 STC15F2K60S2 系列单片机的 IDATA 大小为 256 字节（00-7F 的 DATA 和 80H-FFH 的 IDATA），但由于 STC15F2K60S2 在 RAM 的最后 17 个字节含有写入 ID 号以及相关的测试参数，若用户在程序中需要使用这一部分数据，则一定不要将 IDATALEN 定义为 256。

电子工业出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

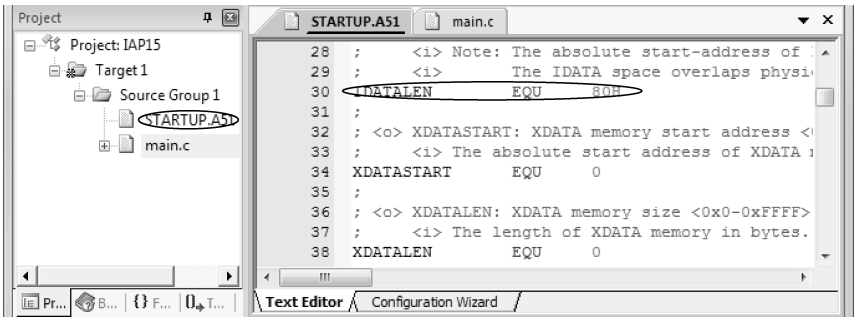



图 2.17　宏定义“IDATALEN”

2）创建仿真芯片 IAP15F2K61S2

打开 STC-ISP，选择正确的单片机型号（IAP15F2K61S2）和串口号（USB-SERIAL CH341A（COM3）），在“Keil 仿真设置”标签中单击“将 IAP15F2K61S2/IAP15L2K61S2 设置为仿真芯片”按钮，界面右下方显示“正在检测目标单片机”，按下开发板上的电源开关，STC-ISP 检测到单片机后开始下载仿真程序，当程序下载完成后仿真器便制作完成，如图 2.18 所示。

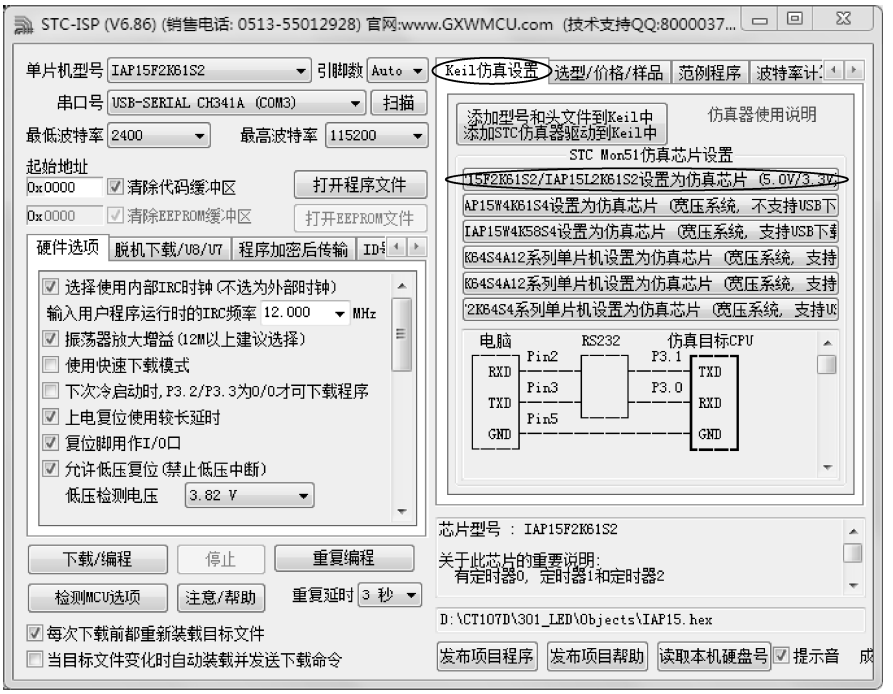



图 2.18　创建仿真芯片

3）硬件仿真驱动选择

在 Keil 中打开“301_LED”工程，在目标选项对话框中选择“Debug”标签，选择右侧的硬件仿真“Use:”，从仿真驱动下拉列表中选择“STC Monitor-51 Driver”，选择“Run to main()”选项，单击“Settings”按钮打开目标设置对话框，选择串口（COM3）和波特率（115200），单击“OK”按钮关闭目标设置和目标选项对话框，如图 2.19 所示。

4）开始仿真调试

编译工程，编译正确后，单击“Debug”菜单下的“Start/Stop Debug Session”菜单项或单击文件工具栏中的“Start/Stop Debug Session”按钮，将编译结果下载到 MCU，然后进入调试界面，如图 2.20 所示。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

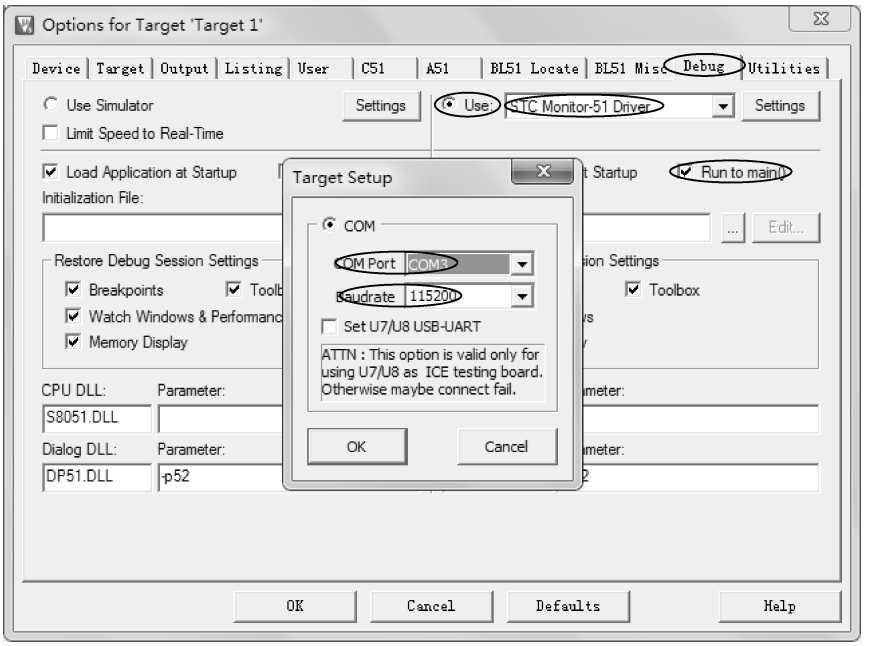



图 2.19　硬件仿真驱动选择

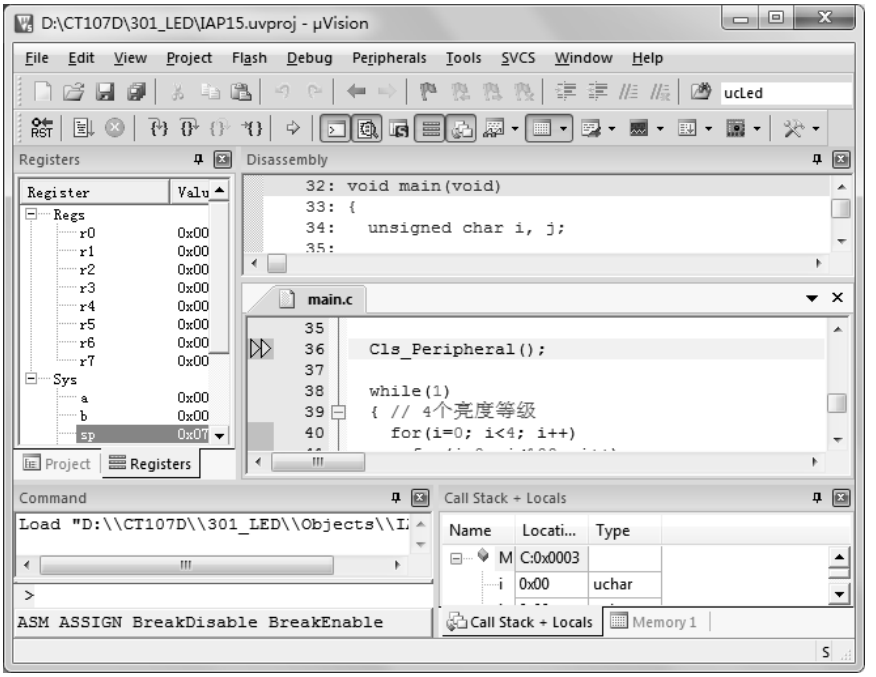



图 2.20　调试界面

调试界面中的调试工具栏如图 2.21 所示，其中包含调试用的工具按钮。

图 2.21　调试工具栏

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

单击“Debug”菜单下的“Run”菜单项或单击调试工具栏中的“Run”按钮运行程序，单片机竞赛实训平台上的 L1～L8 的不同亮度显示。

STC-ISP 除了具有程序下载和仿真设置外，还有其他实用功能，例如串口助手、波特率计算器、定时器计算器和软件延时计算器等，这些功能的使用将在后续章节中介绍。

## 2.2 嵌入式设计与开发项目开发环境与工具

嵌入式设计与开发项目开发环境包括软件开发环境和硬件开发环境，软件开发环境包括软件开发包（SDK）和集成开发环境（IDE）等，硬件开发环境包括国信长天嵌入式竞赛实训平台和嵌入式资源扩展板等（参见附录 B 和附录 C）。

### 2.2.1 软件开发包（SDK）

STM32 软件开发包是 STM32 系统设计的基础，经历了下列 3 个阶段。

- 固件库（FirmWare Library）：FWLib V0.3（2007）~ V2.0.3（2008）
- 标准外设库（Standard Peripherals Library）：SPLib V3.0.0（2009）~ V3.5.0（2011）
- STM32Cube 固件包：HAL V1.0.0（2014）~ V1.6.0（2017）

本书以标准外设库 SPLib 为例介绍标准外设库的组成和使用。

STM32F10x 的标准外设库结构如图 2.22 所示。

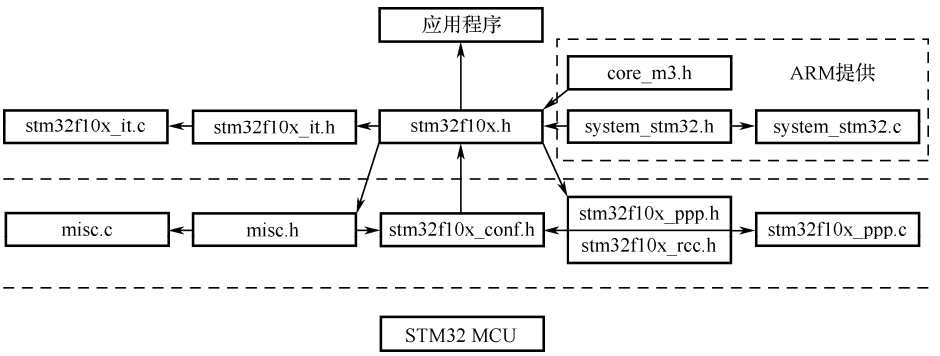



图 2.22　STM32F10x 标准外设库结构

SPLib V3.5.0 主要包括下列内容：

- ARM CM3 CMSIS（Cortex Microcontroller Software Interface Standard：Cortex 微控制器软件接口标准）V1.30：core_cm3.h 和 core_cm3.c
- STM3210x CMSIS V3.5.0：startup_stm32f10x_*.s、stm32f10x.h、system_stm32f10x.h 和 system_stm32f10x.c
- STM32F10x_StdPeriph_Driver V3.5.0：misc.h、misc.c、stm32f10x_*.h 和 stm32f10x_*.c
- STM32F10x_StdPeriph_Examples V3.5.0：外设例程
- STM32F10x_StdPeriph_Template V3.5.0：工程模板
- STM32_EVAL V4.5.0：评估板支持包

SPLib V3.5.0 支持的集成开发环境如下：

- RealView Microcontroller Development Kit（MDK-ARM）V4.13
- IAR Embedded Workbench for ARM（EWARM）V5.50.5
- Atollic TrueSTUDIO STM32（TrueSTUDIO）V1.4.0



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

- Hitex IDE/Debugger（HiTOP）V5.40.0051
- Raisonance IDE（RIDE）V7.30.10

### 2.2.2 集成开发环境（IDE）

从 SPLib V3.5.0 支持的集成开发环境可以看出：STM32 的集成开发环境有 MDK-ARM、EWARM、TrueSTUDIO、HiTOP 和 RIDE 等，本书以 MDK-ARM 为例介绍集成开发环境的组成和使用。

MDK-ARM 是 ARM 收购 Keil 后推出的 ARM MCU 开发工具，是 Keil 集成开发环境μVision 和 ARM 高效编译工具 RVCT（RealView Complie Tools）的完美结合。

MDK-ARM 经历了下列几个阶段：

- DK-ARM V1.0 ~ V1.4
- Keil Development Tools for ARM V1.5
- Keil Development Suite for ARM V2.00 ~ V2.42
- RealView Microcontroller Development Kit V2.50，V3.00 ~ V3.80，V4.00 ~ V4.20
- Microcontroller Development Kit V4.21 ~ V4.73，V5.00 ~ V5.25

早期版本的 MDK-ARM 内嵌软件开发包，如 RVMDK V4.12 内嵌 FWLib V2.0.1，MDK V4.73 内嵌 SPLib V3.5.0。从 MDK-ARM V5.00 开始，软件开发包以 STM32Cube 固件包的形式单独发布，如 stm32cubef1.zip。

考虑到国信长天嵌入式竞赛实训平台上的 Coocox 调试器只能在 MDK-ARM V4.xx 环境下工作，本书以 MDK-ARM V4.73 为例介绍 MDK-ARM 的组成和使用。

MDK-ARM V4.73 的安装包是 MDK473.exe，安装完成后下列文件夹包含 SPLib V3.5.0：

- C:\Keil\ARM\CMSIS\Include：core_cm3.h
- C:\Keil\ARM\INC\ST\STM32F10x：stm32f10x.h、system_stm32f10x.h
- C:\Keil\ARM\Startup\ST\STM32F10x：startup_stm32f10x_*.s、system_stm32f10x.c
- C:\Keil\ARM\RV31\LIB\ST\STM32F10x_StdPeriph_Driver\inc：misc.h、stm32f10x_*.h
- C:\Keil\ARM\RV31\LIB\ST\STM32F10x_StdPeriph_Driver\src：misc.c、stm32f10x_*.c

为了方便使用，可以将竞赛资源包“液晶驱动参考例程”文件夹中的“CT117E-LCD”文件夹复制粘贴到 D 盘，将 D 盘文件夹“CT117E-LCD”更名为“CT117E”，将“Sre”文件夹中的“main.c”剪切粘贴到“Project”文件夹，用 Keil V4.73 打开“Project”文件夹中的“CT117E-LCD”工程，在“Project”下将组名“LCD_Driver”更改为“Driver”，右击“User”组中的“main.c”，在弹出的菜单中选择“Options for File 'main.c'...”菜单项打开“Options for File 'main.c'”对话框，删除“Path”中的“..\Sre\”，单击“Build”按钮在“Output”文件夹中生成“CT117E-LCD.axf”文件。

也可以使用 Keil 新建模板工程，具体步骤如下：

（1）在 D 盘新建“CT117E”文件夹，在“CT117E”文件夹中新建“Inc”“Output”“Project”和“Sre”4 个文件夹，将 SPLib V3.5.0 中的“Libraries”复制粘贴到“CT117E”文件夹，其中包含“CMSIS”和“STM32F10x_StdPeriph_Driver”两个文件夹。

（2）双击桌面的 Keil 图标运行 Keil 程序，选择“Project”>“New μVision Project”菜单项打开新建工程对话框，选择“D:\CT117E\Project”文件夹，在“文件名”中输入工程文件名“STM32”，如图 2.23 所示，单击“保存”按钮关闭新建工程对话框。

（3）在“Select Device”对话框 CPU 标签的“Data base”下选择“STMicroelectronics”公司的“STM32F103RB”器件，如图 2.24 所示。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

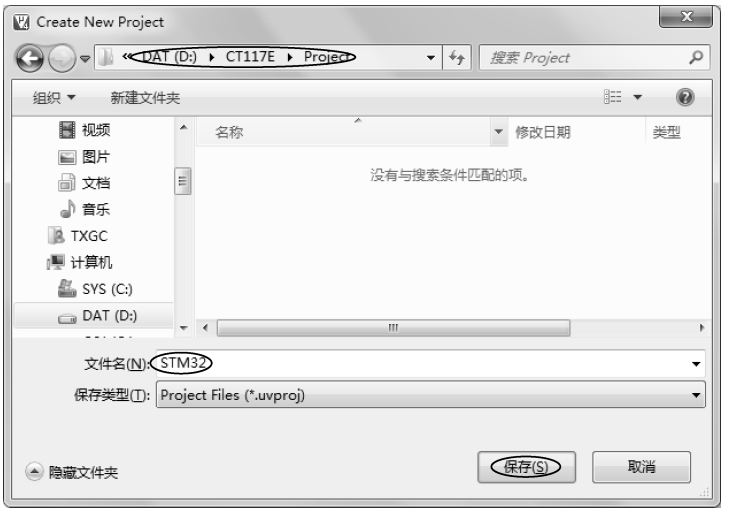



图 2.23　新建工程

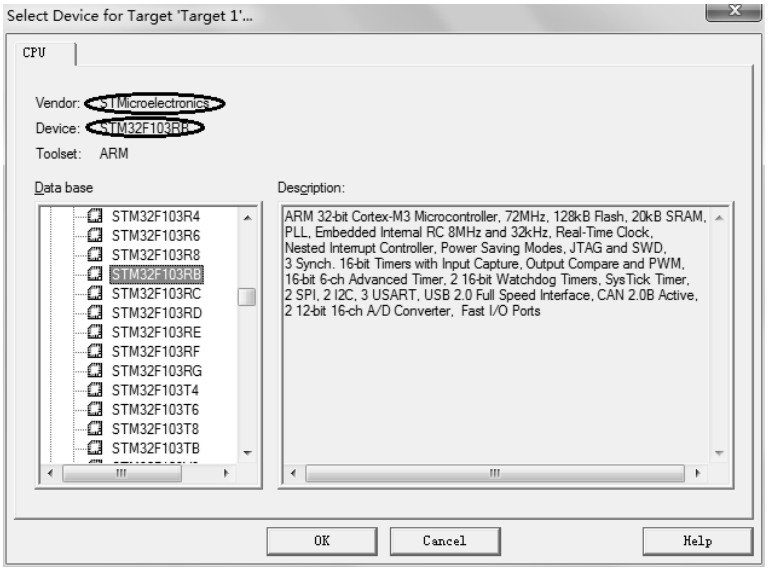



图 2.24　选择器件

单击“OK”按钮关闭选择器件对话框，单击“是”按钮复制 startup_stm32f10x_md.s 到工程文件夹并添加文件到工程（Copy ‘startup_stm32f10x_md.s’ to Project Folder and Add File to Project）。

（4）在“Project”中单击“Target 1”前的加号打开目标，将“Source Group 1”更名为“Startup”，单击“Startup”前的加号打开源文件组，其中包含汇编语言启动代码源文件 startup_stm32f10x_md.s。

（5）在“Project”中右击“Target 1”，在弹出的菜单中选择“Add Group...”菜单项，添加“New Group”，将“New Group”更名为“CMSIS”，双击“CMSIS”打开“Add Files to Group‘CMSIS’”对话框，选择“CT117E”→“Libraries”→“CMSIS”→“CM3”→“DeviceSupport”→“ST”→“STM32F10x”，将下列文件添加到“CMSIS”组：

- system_stm32f10x.c：Cortex-M3 器件外设访问层系统源文件（包含 SystemInit()函数）

添加组“Library”、“Driver”和“User”。

（6）单击生成工具栏中的“Target Option”按钮 打开目标选项对话框，选择“Output”标签，单击“Selecte Folder for Objects...”按钮打开浏览文件夹对话框，选择“CT117E”→“Output”



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

文件夹，单击“OK”关闭浏览文件夹对话框，如图 2.25 所示。

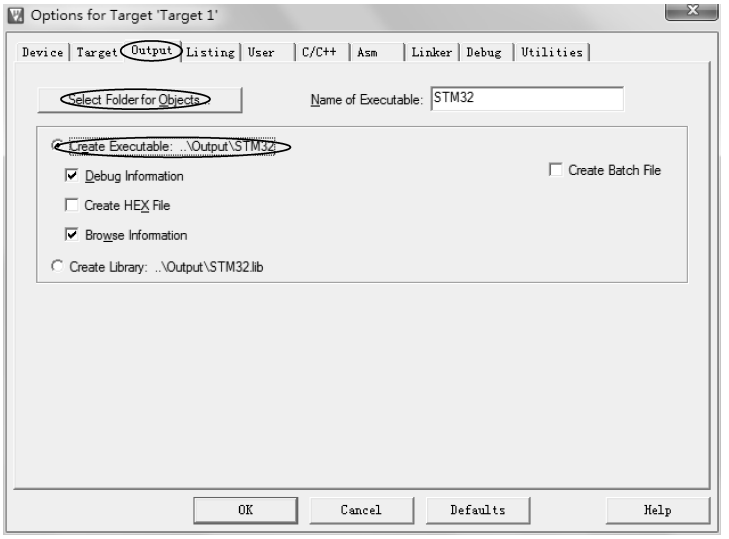



图 2.25　选择输出文件夹

（7）在目标选项对话框中选择“C/C++”标签，在“Preprocessor Symbols”下的“Define”文本框中输入“STM32F10X_MD，USE_STDPERIPH_DRIVER”，单击“Include Paths”文本框右边的…按钮选择下列包含路径，如图 2.26 所示。

- ..\Libraries\CMSIS\CM3\DeviceSupport\ST\STM32F10x
- ..\Libraries\STM32F10x_StdPeriph_Driver\inc
- ..\Inc

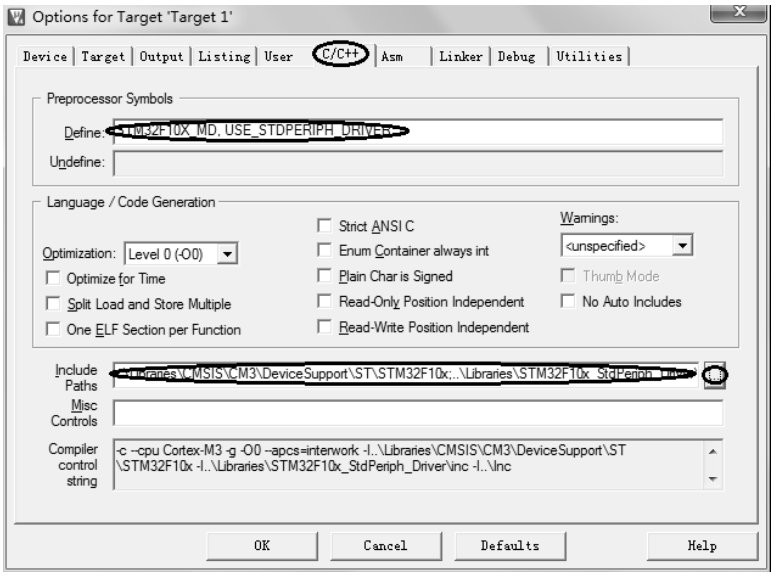



图 2.26　C/C++编译设置

## 2.2.3　Coocox 调试器设置

使用国信长天嵌入式竞赛实训平台板载 Coocox 调试器时，需要安装调试器驱动程序和调试器插件程序，还要对调试器进行设置。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

（1）安装调试器驱动程序：实训平台使用双 USB UART 转换芯片 FT2232D 作为板载调试器，将实训平台通过调试器 USB 插座 CN2 与 PC 相连，PC 提示安装 FT2232 驱动程序，安装完成后显示驱动程序软件安装对话框，如图 2.27 所示。

图 2.27　驱动程序软件安装对话框

设备管理器中出现 USB 设备 USB Serial Converter A/B 和 COM 端口 USB Serial Port (COM22)，不同的 PC 设备号可能不同，如图 2.28 所示。

（a）USB 设备　　（b）COM 端口

图 2.28　调试器 USB 设备和 COM 端口

**注意**：记住 COM 端口号，后面的串行通信要用到。

（2）安装调试器插件程序：运行调试器 Keil 插件安装程序 CoMDKPlugin-1.3.1.exe，将插件程序安装到默认文件夹 C:\Keil。

（3）在 Keil 中打开模板工程，单击生成工具栏中的“Target Option”按钮 打开目标选项对话框，选择“Debug”标签，选择“Use”调试器并从下拉列表中选择“CooCox Debugger”（如果没有 CooCox Debugger，需重新安装调试器插件程序），选中“Run to main()”选项，如图 2.29 所示。

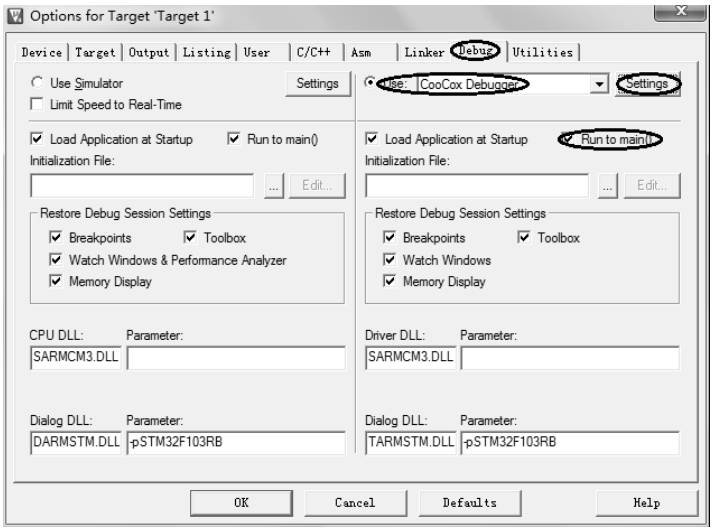



图 2.29　调试器设置

（4）单击“Settings”按钮打开驱动设置对话框，确认“Debug”标签中的“Adapter”为 Colink，“Port”为 JTAG，“IDCODE”为 0x3BA00477，“Device Name”为 ARM CoreSight JTAG-DP，如图 2.30 所示。



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

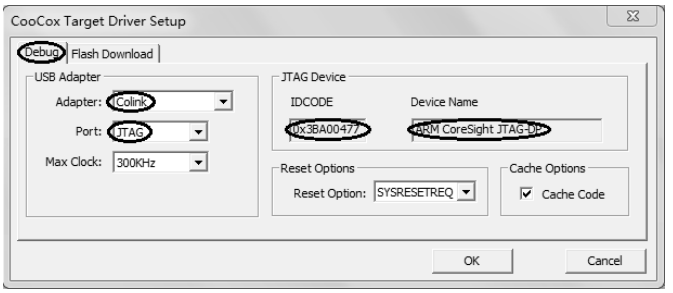



图 2.30　驱动设置-调试设置

（5）选择“Flash Download”标签，单击“Add”按钮打开“Add Programming Algorithm”对话框，选择“STM32F10x Med-density Flash”，如图 2.31 所示。

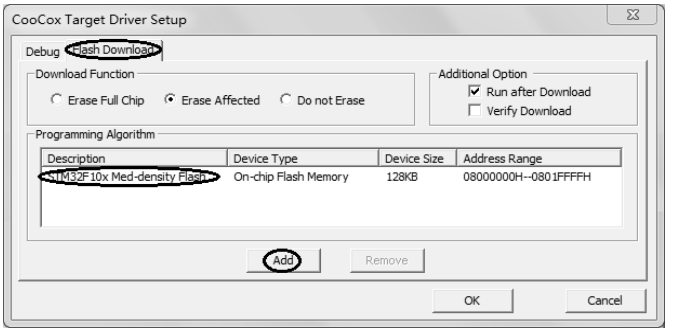



图 2.31　驱动设置-下载设置

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

# 第 3 章　单片机设计与开发项目模块程序设计

本章介绍各模块程序设计，包括头文件、源文件和示例程序。为了使用方便，头文件和源文件存放在“D:\CT107D\Driver”文件夹中，示例程序存放在各自的工程文件夹中。

单片机设计与开发项目模块程序设计包括下列内容：

（1）LED 程序设计。

（2）中断程序设计。

（3）定时器程序设计。

（4）数码管程序设计。

（5）矩阵键盘程序设计。

（6）串口程序设计。

（7）DS1302 时钟芯片驱动程序设计。

（8）DS18B20 温度传感器驱动程序设计。

（9）PCF8591 ADC/DAC 驱动程序设计。

（10）AT24C02 存储器驱动程序设计。

（11）超声波传感器驱动程序设计。

（12）频率测量程序设计。

## 3.1　LED 程序设计

国信长天单片机竞赛实训平台的 LED 电路图如图 3.1 所示。

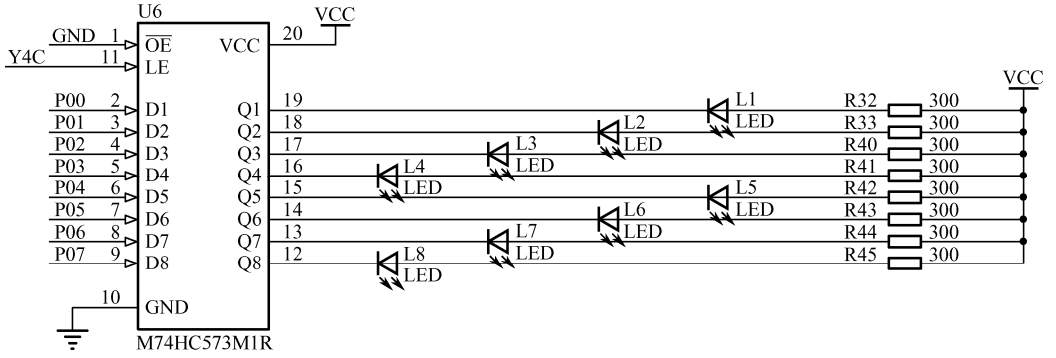



图 3.1　国信长天单片机竞赛实训平台 LED 电路图

电路采用 M74HC573M1R 锁存器对单片机 P0 口的输出信号进行缓冲，并增强信号驱动能力进而驱动 LED 发光。其中 U6 的锁存输入 $Y4C=\overline{Y4+WR}=\overline{Y4}\cdot\overline{WR}$，Y4 是译码器输入 P27～P25 为 100 时的有效输出，WR 可接地（IO 模式，J13-2 和 J13-3 相连），也可以接 P36（MM 模式，J13-2 和 J13-1 相连），相对而言，MM 模式简单一些（必须包含 absacc.h）。

基于国信长天单片机竞赛实训平台的 LED 亮度控制程序设计如下：

```
// 使用程序前，将 J13 调整为 IO 模式（2、3 脚短接）
#include "reg52.h"
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
// #include "absacc.h"
// 关闭外设
void Cls_Peripheral(void)
{ // IO 模式（J13-2 和 J13-3 相连）
  P0 = 0xFF;
  P2 = P2 & 0x1F | 0x80;          // P27～P25 清零，再定位 Y4C
  P2 &= 0x1F;                     // P27～P25 清零
  P0 = 0;
  P2 = P2 & 0x1F | 0xA0;          // P27～P25 清零，再定位 Y5C
  P2 &= 0x1F;                     // P27～P25 清零
//XBYTE[0x8000] = 0xFF;           // MM 模式（J13-2 和 J13-1 相连）
//XBYTE[0xA000] = 0;
}
// LED 显示
void Led_Disp(unsigned char ucLed)
{ // IO 模式（J13-2 和 J13-3 相连）
  P0 = ~ucLed;
  P2 = P2 & 0x1F | 0x80;          // P27～P25 清零，再定位 Y4C
  P2 &= 0x1F;                     // P27～P25 清零
//XBYTE[0x8000] = ~ucLed;         // MM 模式（J13-2 和 J13-1 相连）
}
// 延时函数（最小约 1ms@12MHz）
void Delay(unsigned int num)
{
  unsigned int i;
  while(num--)
    for(i=0; i<628; i++);
}
// 主函数
void main(void)
{
  unsigned char i, j;

  Cls_Peripheral();

  while(1)
  { // 4 个亮度等级
    for(i=0; i<4; i++)
      for(j=0; j<100; j++)
      {
        Led_Disp(0xff);
        Delay(i+1);
        Led_Disp(0);
        Delay(4-i);
      }
  }
}
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

**注意**：运行程序时，将 J13 调整为 IO 模式（2、3 脚短接）。

## 3.2 中断程序设计

IAP15F2K61S2 系列单片机提供了 14 个中断请求源，它们分别是：外部中断 0（INT0）、定时器 0 中断、外部中断 1（INT1）、定时器 1 中断、串口 1 中断、ADC 中断、低压检测（LVD）中断、CCP/PWM/PCA 中断、串口 2 中断、SPI 中断、外部中断 2（INT2）、外部中断 3（INT3）、定时器 2 中断，以及外部中断 4（INT4）。除外部中断 2（INT2）、外部中断 3（INT3）、定时器 2 中断和外部中断 4（INT4）固定是最低优先级中断外，其他的中断都具有两个中断优先级，可实现 2 级中断服务程序嵌套。本节重点介绍外部中断 0（INT0）的相关内容及使用。

外部中断 0（INT0）和外部中断 1（INT1）既可上升沿触发，又可下降沿触发，两个外部中断的标志位是位于寄存器 TCON 中的 IE0/TCON.1 和 IE1/TCON.3。当外部中断服务程序响应后，中断标志位 IE0 和 IE1 会自动被清零。TCON 寄存器中的 IT0/TCON.0 和 IT1/TCON.2 决定了外部中断 0 和 1 是上升沿触发还是下降沿触发。如果 ITx=0（x=0,1），那么系统在 INTx（x=0,1）脚探测到上升沿或下降沿后均可产生外部中断。如果 ITx=1（x=0,1），那么系统在 INTx（x=0,1）脚探测到下降沿后可产生外部中断。外部中断 0（INT0）和外部中断 1（INT1）还可以用于将单片机从掉电模式唤醒。

定时器 0 和 1 的中断请求标志位是 TF0 和 TF1。外部中断 2（INT2）、外部中断 3（INT3）及外部中断 4（INT4）都只能下降沿触发。其他中断触发方式详见芯片数据手册。

部分中断源信息如表 3.1 所示。

表 3.1 部分中断源信息

| 中 断 源 | 中断向量地址 | 中 断 号 | 中断优先级设置 | 优先级 1（最高） | 中断请求标志位 | 中断允许控制位 |
|---|---|---|---|---|---|---|
| INT0（外部中断 0） | 0003H | 0 | PX0 | 1 | IE0 | EX0/EA |
| Timer 0 | 000BH | 1 | PT0 | 1 | TF0 | ET0/EA |
| INT1（外部中断 1） | 0013H | 2 | PX1 | 1 | IE1 | EX1/EA |
| Timer1 | 001BH | 3 | PT1 | 1 | TF1 | ET1/EA |
| UART1 | 0023H | 4 | PS | 1 | RI+TI | ES/EA |

### 1）中断允许寄存器 IE

IAP15F2K61S2 系列单片机 CPU 对中断源的开放或屏蔽，每一个中断源是否被允许中断，是由内部的中断允许寄存器 IE（IE 为特殊功能寄存器，它的字节地址为 A8H）控制的，其格式如表 3.2 所示。

表 3.2 中断允许寄存器 IE（可位寻址）格式

| 名称 | 地址 | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|---|---|---|---|---|---|---|---|---|---|
| IE | A8H | EA | ELVD | EADC | ES | ET1 | EX1 | ET0 | EX0 |

EA：CPU 的总中断允许控制位，EA=1，CPU 开放中断；EA=0，CPU 屏蔽所有的中断申请。EA 的作用是使中断允许形成多级控制。即各中断源首先受 EA 控制；其次还受各中断源自己的中断允许控制位控制。

ELVD：低压检测中断允许位，ELVD=1，允许低压检测中断；ELVD=0，禁止低压检测中断。

EADC：ADC 中断允许位，EADC=1，允许 ADC 中断；EADC=0，禁止 ADC 中断。



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

ES：串行口 1 中断允许位，ES=1，允许串行口 1 中断；ES=0，禁止串行口 1 中断。

ET1：定时/计数器 T1 的溢出中断允许位，ET1=1，允许 T1 中断；ET1=0，禁止 T1 中断。

EX1：外部中断 1 中断允许位，EX1=1，允许外部中断 1 中断；EX1=0，禁止外部中断 1 中断。

ET0：T0 的溢出中断允许位，ET0=1 允许 T0 中断；ET0=0 禁止 T0 中断。

EX0：外部中断 0 中断允许位，EX0=1，允许外部中断 0 中断；EX0=0，禁止外部中断 0 中断。

2）中断优先级控制寄存器 IP

STC15 系列单片机通过设置特殊功能寄存器（IP 和 IP2）中的相应位，可将部分中断设有两个中断优先级，并设有两条基本规则：

（1）低优先级中断可被高优先级中断所中断，反之不能。

（2）任何一种中断（不管是高级还是低级），一旦得到响应，就不会再被它的同级中断所中断。

中断优先级控制寄存器 IP 的格式如表 3.3 所示。

表 3.3　中断优先级控制寄存器 IP（可位寻址）的格式

| 名称 | 地址 | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|---|---|---|---|---|---|---|---|---|---|
| IP | B8H | PPCA | PLVD | PADC | PS | PT1 | PX1 | PT0 | PX0 |

PT1：定时器 1 中断优先级控制位。

当 PT1=0 时，定时器 1 中断为最低优先级中断（优先级 0）；

当 PT1=1 时，定时器 1 中断为最高优先级中断（优先级 1）。

PX1：外部中断 1 优先级控制位。

当 PX1=0 时，外部中断 1 为最低优先级中断（优先级 0）；

当 PX1=1 时，外部中断 1 为最高优先级中断（优先级 1）。

PT0：定时器 0 中断优先级控制位。

当 PT0=0 时，定时器 0 中断为最低优先级中断（优先级 0）；

当 PT0=1 时，定时器 0 中断为最高优先级中断（优先级 1）。

PX0：外部中断 0 优先级控制位。

当 PX0=0 时，外部中断 0 为最低优先级中断（优先级 0）；

当 PX0=1 时，外部中断 0 为最高优先级中断（优先级 1）。

3）定时器/计数器控制寄存器 TCON

TCON 为定时器/计数器 T0 和 T1 的控制寄存器，同时也锁存 T0 和 T1 溢出中断源和外部请求中断源等，TCON 格式如表 3.4 所示。

表 3.4　定时器/计数器中断控制寄存器 TCON（可位寻址）

| 名称 | 地址 | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|---|---|---|---|---|---|---|---|---|---|
| TCON | 88H | TF1 | TR1 | TF0 | TR0 | IE1 | IT1 | IE0 | IT0 |

TF1：T1 溢出中断标志。T1 被允许计数以后，从初值开始加 1 计数。当产生溢出时由硬件置“1”TF1，向 CPU 请求中断，一直保持到 CPU 响应中断时，才由硬件清零（也可由查询软件清零）。

TR1：定时器 1 的运行控制位。

TF0：T0 溢出中断标志。T0 被允许计数以后，从初值开始加 1 计数，当产生溢出时，由硬件置“1”TF0，向 CPU 请求中断，一直保持 CPU 响应该中断时，才由硬件清零（也可由查询软件清零）。

TR0：定时器 0 的运行控制位。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

IE1：外部中断 1（INT1/P3.3）中断请求标志。IE1=1，外部中断向 CPU 请求中断，当 CPU 响应该中断时由硬件清零 IE1。

IT1：外部中断 1 中断源类型选择位。IT1=0，INT1/P3.3 引脚上的上升沿或下降沿信号均可触发外部中断 1。IT1=1，外部中断 1 为下降沿触发方式。

IE0：外部中断 0（INT0/P3.2）中断请求标志。IE0=1，外部中断 0 向 CPU 请求中断，当 CPU 响应外部中断时，由硬件清零 IE0。

IT0：外部中断 0 中断源类型选择位。IT0=0，INT0/P3.2 引脚上的上升沿或下降沿均可触发外部中断 0。IT0=1，外部中断 0 为下降沿触发方式。

中断函数的一般形式为：

```
void 函数名() [interrupt n] [using m]
```

关键字 interrupt 后面的 n 是中断号，编译器从 8n+3 处产生中断入口地址，具体的中断号 n 和中断向量取决于所采用的中断源，其中外部中断 0 的中断号为 0。

关键字 using 专门用来选择单片机中不同的工作寄存器组。using 后面的 m 是一个 0～3 的常整数，分别选中 4 个不同的工作寄存器组。定义中断函数时 using 是一个选项，如果不用该选项，则由编译器自动选择一个寄存器组进行绝对寄存器组访问。

应该特别注意，在任何情况下都不能直接调用中断函数，因为它不能进行参数传递，也没有返回值。

基于国信长天单片机竞赛实训平台的外部中断程序设计如下：

```
// 使用程序前，将跳线 J5 调整为 BTN 模式（2、3 引脚短接使第一列按键一端接地）
// 将跳线 J13 调整为 IO 模式（2、3 引脚短接使 U25 或非门输入信号 WR 接地）
// 按下 P32-INT0 引脚对应的按键 S5，L1 状态翻转
// 按下 P33-INT1 引脚对应的按键 S4，L2 状态翻转
#include "reg52.h"
unsigned char ucLed;
// 关闭外设
void Cls_Peripheral(void)
{
  P0 = 0xFF;
  P2 = P2 & 0x1F | 0x80;              // P27～P25 清零，再定位 Y4C
  P2 &= 0x1F;                         // P27～P25 清零
  P0 = 0;
  P2 = P2 & 0x1F | 0xA0;              // P27～P25 清零，再定位 Y5C
  P2 &= 0x1F;                         // P27～P25 清零
}
// 系统初始化
void Sys_Init(void)
{
  EX0 = 1;
  IT0 = 1;                            // 边沿触发方式（下降沿)
  EX1 = 1;
  IT1 = 1;                            // 边沿触发方式（下降沿)
  EA = 1;                             // 开启总中断
}
```



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
// 主函数
void main(void)
{
  Cls_Peripheral();
  Sys_Init();
  while(1);
}
//外部中断 0 中断服务函数
void isr_intr_0(void) interrupt 0
{
  ucLed ^= 1;
  P0 = ~ucLed;
  P2 = P2 & 0x1F | 0x80;              // P27～P25 清零，再定位 Y4C
  P2 &= 0x1F;                         // P27～P25 清零
}
//外部中断 1 中断服务函数
void isr_intr_1(void) interrupt 2
{
  ucLed ^= 2;
  P0 = ~ucLed;
  P2 = P2 & 0x1F | 0x80;              // P27～P25 清零，再定位 Y4C
  P2 &= 0x1F;                         // P27～P25 清零
}
```

中断程序的实现方法是：在“D:\CT107D”文件夹中，将“201_LED”文件夹复制并重命名为“202_INT”文件夹，进入“202_INT”文件夹，双击工程文件“IAP15”打开工程，将“main.c”的内容替换为上述程序，编译并调试程序。

**注意：**运行程序时，将 J5 调整为 BTN 模式（2、3 脚短接），将 J13 调整为 IO 模式（2、3 脚短接）。由于没有消抖，一次按键可能造成 LED 翻转两次。

## 3.3 定时器程序设计

IAP15F2K61S2 单片机内部集成了 3 个 16 位的定时/计数器 T0、T1 和 T2，其核心部件是一个加法计数器（TH 和 TL），对输入脉冲进行计数。若计数脉冲来自系统时钟，则为定时器方式；若计数脉冲来自 T0（P34）、T1（P35）或 T2（P31）引脚，则为计数器方式。

定时/计数器 T0 和 T1 具有四种工作方式，由特殊功能寄存器 TMOD 中的 M1 和 M0 位决定，如表 3.5 所示。

表 3.5　定时/计数器 T0 和 T1 的四种工作方式

| M1　M0 | 工 作 方 式 | 功 能 说 明 |
|---|---|---|
| 0　0 | 方式 0 | 16 位自动重装初值的定时/计数器 |
| 0　1 | 方式 1 | 16 位的定时/计数器 |
| 1　0 | 方式 2 | 8 位自动重装初值的定时/计数器 |
| 1　1 | 方式 3 | T0 被分成两个 8 位的定时/计数器，T1 停止计数 |

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

定时/计数器 T0 和 T1 的工作方式 1 至工作方式 3 与传统的 8051 类似，但是工作方式 0 为 16 位自动重装方式，这一点与传统的 8051 不同。另外工作方式 0 和工作方式 2 还可以作为可编程时钟输出源使用。定时/计数器 T0 工作方式 0 的逻辑结构如图 3.2 所示。

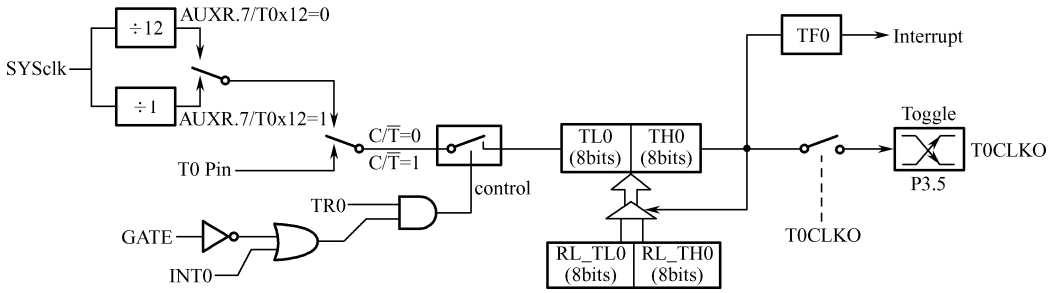



图 3.2　定时/计数器 T0 工作方式 0 的逻辑结构

定时/计数器 T2 的工作方式固定为 16 位自动重装初值方式，逻辑结构如图 3.3 所示，T2 除了可用作定时器，还可以用作串行口波特率发生器或可编程时钟输出源。

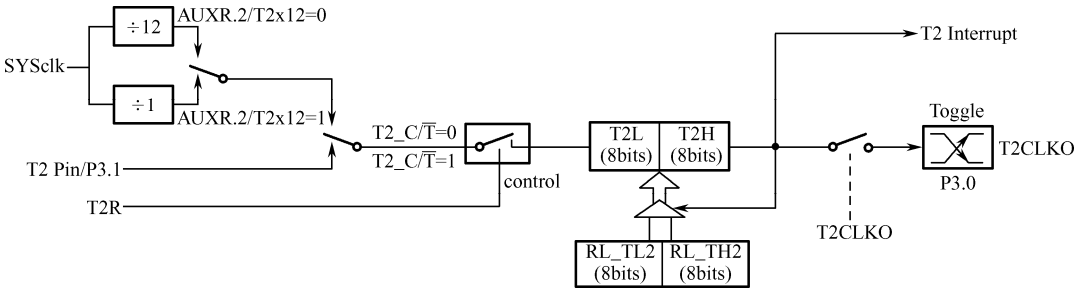



图 3.3　定时/计数器 T2 的逻辑结构

定时器输入脉冲方式可以是系统时钟脉冲 fsys，简称 1T 方式；也可以对系统时钟 fsys 进行 12 分频之后作为定时输入脉冲，简称 12T 方式，12T 方式与传统的定时器方式相同。采用 1T 还是 12T 方式，由特殊功能寄存器 AUXR 决定。当 AUXR 寄存器中的 T0x12、T1x12 和 T2x12 为 0 时，采用 12T 方式；当 T0x12、T1x12 和 T2x12 为 1 时，采用 1T 方式。

计数方式下从外部引脚输入的脉冲，必须保证脉冲电平在变化之前至少被采样一次。

IAP15F2K61S2 单片机与定时/计数器相关的特殊功能寄存器如表 3.6 所示。

**表 3.6　定时/计数器的特殊功能寄存器**

| 寄存器名 | 地址 | MSB　复位地址及位名称　LSB | | | | | | | | 复位值 |
|---|---|---|---|---|---|---|---|---|---|---|
| TH0 | 8CH | 定时/计数器 T0 的高 8 位 | | | | | | | | 0000 0000 |
| TL0 | 8AH | 定时/计数器 T0 的低 8 位 | | | | | | | | 0000 0000 |
| TH1 | 8DH | 定时/计数器 T1 的高 8 位 | | | | | | | | 0000 0000 |
| TL1 | 8BH | 定时/计数器 T1 的低 8 位 | | | | | | | | 0000 0000 |
| T2H | D6H | 定时/计数器 T2 的高 8 位 | | | | | | | | 0000 0000 |
| H2L | D7H | 定时/计数器 T2 的低 8 位 | | | | | | | | 0000 0000 |
| TCON | 88H | TF1 | TR1 | TF0 | TR0 | IE1 | IT1 | IE0 | IT0 | 0000 0000 |
| TMOD | 89H | GATE | C/T | M1 | M0 | GATE | C/T | M1 | M0 | 0000 0000 |
| IE | A8H | EA | ELVD | EADC | ES | ET1 | EX1 | ET0 | EX0 | 0000 0000 |
| IE2 | AFH | - | - | - | - | - | EET2 | ESP1 | ES2 | 0000 0000 |



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

续表

| 寄存器名 | 地址 | MSB 复位地址及位名称 LSB | | | | | | | | 复位值 |
|---|---|---|---|---|---|---|---|---|---|---|
| IP | B8H | PPCA | PLVD | PADC | PS | PT1 | PX1 | PT0 | PX0 | 0000 0000 |
| INT_CLKO | 8FH | - | EX4 | EX3 | EX2 | - | T2CLKO | T1CLKO | T0CLKO | 0000 0000 |
| AUXR | 8EH | T0x12 | T1x12 | UART_M0x6 | T2R | T2_C/T | T2x12 | WXTRAM | S1ST2 | 0000 0000 |

（1）寄存器 TH0、TL0、TH1、TL1、T2H 和 T2L 分别用于装入 T0、T1 和 T2 的初值。

（2）寄存器 TCON 和 TMOD 的功能和用法与传统 8051 相同。

（3）寄存器 IE 和 IE2 中 ET1、ET0 和 ET2 位分别用于 T1、T0 和 T2 的中断允许或禁止，置 1 允许中断，清零禁止中断。

（4）寄存器 IP 中的 PT1 和 PT0 位分别用于 T1 和 T0 的中断优先级，置 1 为高优先级，清零为低优先级；T2 固定为最低优先级。

（5）寄存器 INT_CLKO 中的 T2_CLKO、T1_CLKO 和 T0_CLKO 位分别用于控制 T2、T1 和 T0 的可编程时钟输出，置 1 允许从 P30、P34 和 P35 引脚输出时钟频率，清零禁止输出时钟频率。

（6）寄存器 AUXR 中的 T0x12、T1x12 和 T2x12 位分别用于控制 T0、T1 和 T2 的 1T 或 12T 定时方式，置 1 采用 1T 方式，清零采用 12T 方式。AUXR 中的 T2R 位用于控制 T2 的启动和停止，置 1 启动 T2，清零停止 T2。AUXR 中的 T2_C/T 位用于控制 T2 的计数或定时方式，置 1 采用计数方式，置 1 采用定时方式。

定时/计数器在进行定时或者计数操作之前要进行初始化配置，通常包括以下几个步骤：

（1）确定工作方式，给方式控制寄存器 TMOD 写入控制字。

（2）确定是采用 1T 还是采用 12T 方式，计算定时/计数器的初值，并将初值写入寄存器 TH 和 TL。

（3）根据需要设置中断控制寄存器 IE 和 IE2 的初值，决定是否开放定时器中断。

（4）根据需要设置寄存器 INT_CLKO 中的 T2_CLKO、T1_CLKO 和 T0_CLKO 位，确定是否采用可编程时钟输出。

（5）置 1 控制寄存器 TCON 和 AUXR 中的 TR0、TR1 和 TR2 位，启动定时/计数器。

在初始化过程中，设置定时器或计数器的初始值，需要进行一些运算。设计数器最大计数值为 $M$（根据不同工作方式，$M$ 可以是 2 的 16 次方或 2 的 8 次方），则初值 $X$ 的计算公式如下：

计数方式：$X=M-$要求的计数值

12T 定时方式：$X=M-$要求的定时值/(12/系统时钟频率)

1T 定时方式：$X=M-$要求的定时值/(1/系统时钟频率)

在定时器方式下，当初值 $X=0$ 时，可以确定最大定时时间，例如当系统时钟频率为 6MHz 时，定时器工作于方式 0，采用 12T 定时方式的最大定时时间为 131.072ms，采用 1T 定时方式的最大定时时间为 10.922ms。如果需要的定时时间大于上述最大定时值，则需要采用中断方式来扩展定时时间。

STC-ISP 是一款不仅可以下载程序还可以实现单片机编程的软件。它可以将硬件设备的芯片重新编辑，利用 ISP 功能，实现数据修改，完成新设备增量控制。其实设备的硬件数据以及芯片是可以通过软件技术重新设置的，利用软件从单片机上下载编程，并将原版数据显示在编辑区域，方便调试人员添加新的产品数据，提升单片机性能。STC-ISP 具有定时器计算器，只需要设定相关参数便可自动生成 C 代码或 ASM 代码，如图 3.4 所示。

基于国信长天单片机竞赛实训平台，设置系统频率为 12.000MHz，设置定时时长为 1ms，选择定时器 1，选择定时器模式为 16 位自动重载（15 系列），选择定时器时钟为 12T（FOSC/12），

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

即可生成相应的 C 代码。

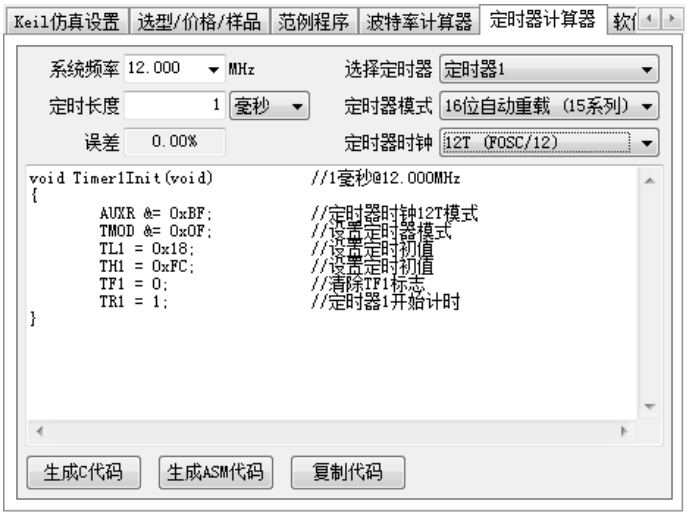



图 3.4　STC-ISP 定时器计算器

定时器头文件“tim.h”内容如下（存放在“D:\CT107D\Driver”文件夹中）：

```
#ifndef __TIM_H
#define __TIM_H
#include "STC15F2K60S2.H"

void Cls_Peripheral(void);
void Led_Disp(unsigned char ucLed);
void Timer1Init(void);
#endif
```

定时器源文件“tim.c”内容如下（存放在“D:\CT107D\Driver”文件夹中）：

```
#include "tim.h"

void Cls_Peripheral(void)
{
  P0 = 0xFF;
  P2 = P2 & 0x1F | 0x80;          // P27～P25 清零，再定位 Y4C
  P2 &= 0x1F;                     // P27～P25 清零
  P0 = 0;
  P2 = P2 & 0x1F | 0xA0;          // P27～P25 清零，再定位 Y5C
  P2 &= 0x1F;                     // P27～P25 清零
}

void Led_Disp(unsigned char ucLed)
{ // IO 模式（J13-2 和 J13-3 相连）
  P0 = ~ucLed;
  P2 = P2 & 0x1F | 0x80;          // P27～P25 清零，再定位 Y4C
  P2 &= 0x1F;                     // P27～P25 清零
//XBYTE[0x8000] = ~ucLed;         // MM 模式（J13-2 和 J13-1 相连）
```



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
}

void Timer1Init(void)            // 1 毫秒@12.000MHz
{
  AUXR &= 0xBF;                  // 定时器时钟 12T 模式
  TMOD &= 0x0F;                  // 设置定时器模式
  TL1 = 0x18;                    // 设置定时初值
  TH1 = 0xFC;                    // 设置定时初值
  TF1 = 0;                       // 清除 TF1 标志
  TR1 = 1;                       // 定时器 1 开始计时
  ET1 = 1;                       // 允许定时器 1 中断
  EA = 1;                        // 允许系统中断
}
```

定时器示例程序“main.c”内容如下：

```
// 使用程序前，将 J13 调整为 IO 模式（2、3 脚短接）
#include "tim.h"

unsigned char ucSec;
unsigned long ulms;

void main(void)
{
  Cls_Peripheral();
  Timer1Init();
  while(1);
}

void Time_1(void) interrupt 3
{
  ulms++;
  if(!(ulms % 1000))
  {
    ucSec++;
    Led_Disp(ucSec);
  }
}
```

定时器程序实现的方法如下：

（1）在“D:\CT107D”文件夹中将“202_INT”文件夹复制并重命名为“203_TIM”文件夹，进入“203_TIM”文件夹，双击工程文件“IAP15”打开工程。

（2）在“Driver”文件夹中新建文件“tim.h”和“tim.c”。

（3）将“tim.c”添加到“Source Group 1”中。

（4）替换“main.c”的内容。

（5）把“..\Driver”添加到包含路径中，如图 3.5 所示。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

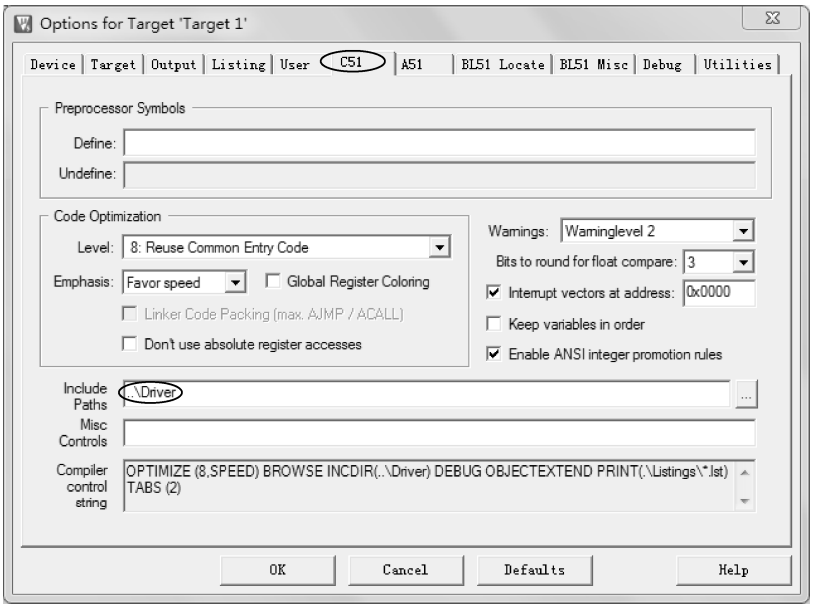



图 3.5　添加包含路径

（6）编译并调试程序。

**注意：**运行程序时，将 J13 调整为 IO 模式（2、3 脚短接）。

## 3.4　数码管程序设计

数码管是一种半导体发光器件，其基本单元是发光二极管。数码管按段数分为七段数码管和八段数码管，八段数码管比七段数码管多一个发光二极管单元（小数点显示）；按能显示多少个“8”可分为 1 位、2 位、4 位等数码管；按发光二极管单元连接方式分为共阳极数码管和共阴极数码管。

共阳极数码管是指将所有发光二极管的阳极接到一起形成公共阳极（COM）的数码管。共阳极数码管在应用时应将公共极 COM 接到+5V，当某一字段发光二极管的阴极为低电平时，相应字段点亮，当某一字段的阴极为高电平时，相应字段不亮。

共阴极数码管是指将所有发光二极管的阴极接到一起形成公共阴极（COM）的数码管，共阴极数码管在应用时应将公共极 COM 接到地线 GND，当某一字段发光二极管的阳极为高电平时，相应字段点亮，当某一字段的阳极为低电平时，相应字段不亮。

数码管要正常显示，就要用驱动电路来驱动数码管的各个段码，从而显示出我们要的数字。根据数码管驱动方式的不同，可以分为静态驱动和动态驱动两类。

（1）静态驱动：也称直流驱动，静态驱动是指每个数码管的每一个段码都由一个单片机的 I/O 端口进行驱动，或者使用如 BCD 码二–十进制译码器进行驱动。静态驱动的优点是编程简单，显示亮度高，缺点是占用 I/O 端口多，如驱动 5 个数码管静态显示则需要 5×8=40 个 I/O 端口来驱动，要知道一个 89C52 单片机可用的 I/O 端口才 32 个，实际应用时必须增加译码驱动器进行驱动，增加了硬件电路的复杂性。

（2）动态驱动：数码管动态显示接口是单片机中应用最为广泛的一种显示方式之一，动态驱动是将所有数码管的 8 个显示笔画“a、b、c、d、e、f、g、dp”的同名端连在一起，另外为每个数码管的公共极 COM 增加位选通控制电路，位选通由各自独立的 I/O 线控制。

当单片机输出字形码时，所有数码管都接收到相同的字形码，但究竟是哪个数码管会显示出



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

字形，取决于单片机对位选通 COM 端电路的控制，所以只要将需要显示的数码管的选通控制打开，该位就显示出字形，没有选通的数码管就不会亮。通过分时轮流控制各个数码管的 COM 端，就使各个数码管轮流受控显示，这就是动态驱动。

在轮流显示过程中，每位数码管的点亮时间为 1～2ms，由于人的视觉暂留现象及发光二极管的余辉效应，尽管实际上各位数码管并非同时点亮，但只要扫描的速度足够快，给人的印象就是一组稳定的显示数据，不会有闪烁感，动态显示的效果和静态显示是一样的，能够节省大量的 I/O 端口，而且功耗更低。

国信长天单片机竞赛实训平台的数码管电路图如图 3.6 所示。

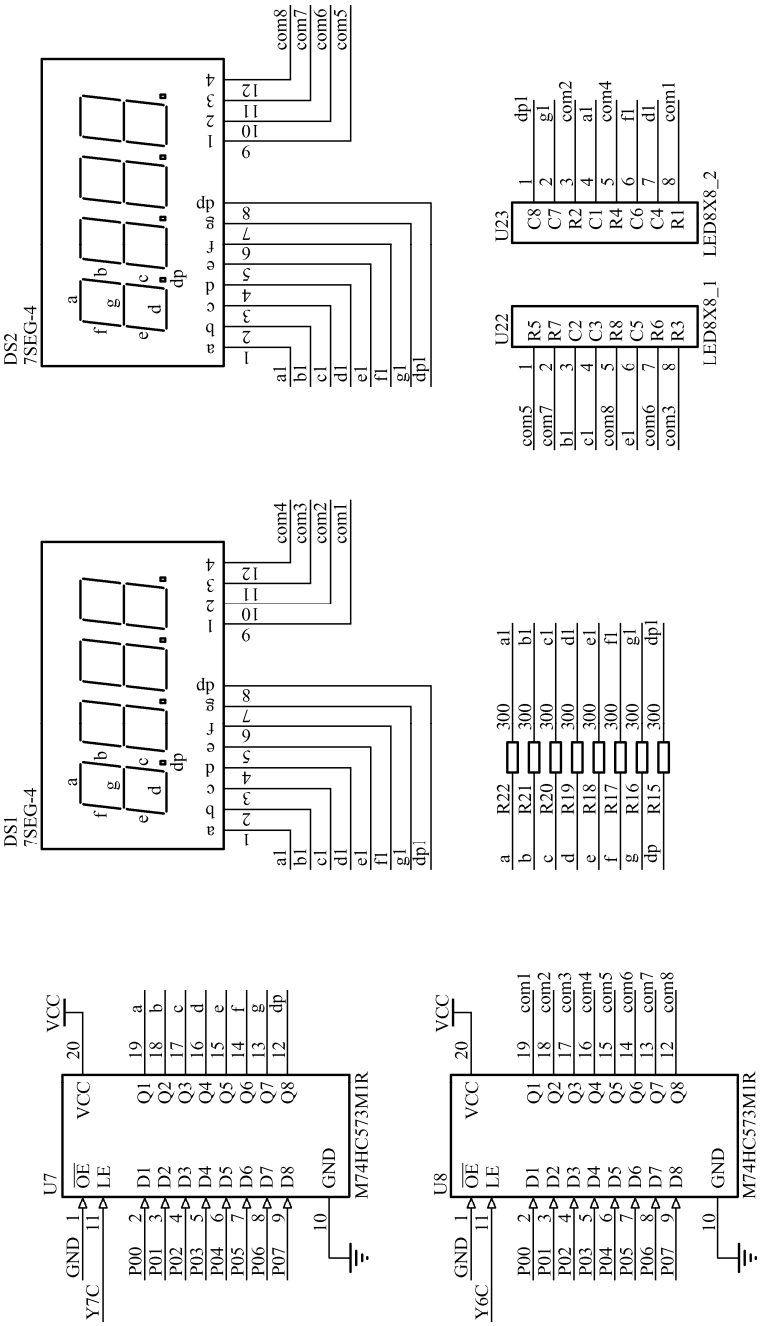



图 3.6　国信长天单片机竞赛实训平台数码管电路图

电子工业出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

8 个数码管采用 2 个 M74HC573M1R 锁存器对单片机 P0 口的输出信号进行锁存，并增强信号驱动能力进而驱动 8 个数码管。其中 U7 的锁存输入 $Y7C=\overline{Y7+WR}=\overline{Y7}\cdot\overline{WR}$ 是段选，U8 的锁存输入 $Y6C=\overline{Y6+WR}=\overline{Y6}\cdot\overline{WR}$ 是位选。和 LED 类似，数码管的操作也有 IO 和 MM 两种模式，这里使用 MM 模式（J13-2 和 J13-1 相连）。

基于国信长天单片机竞赛实训平台的数码管显示程序设计如下。

数码管头文件“seg.h”内容如下（存放在“D:\CT107D\Driver”文件夹中）：

```
#ifndef __SEG_H
#define __SEG_H
#include "STC15F2K60S2.H"

void Seg_Tran(unsigned char *pucSeg_Buf, unsigned char *pucSeg_Code);
void Seg_Disp(unsigned char *pucSeg_Code, unsigned char ucSeg_Pos);
#endif
```

数码管源文件“seg.c”内容如下（存放在“D:\CT107D\Driver”文件夹中）：

```
#include "seg.h"
// 显示转换
void Seg_Tran(unsigned char *pucSeg_Buf, unsigned char *pucSeg_Code)
{
  unsigned char i, j=0, temp;

  for(i=0; i<8; i++, j++)
  {
    switch(pucSeg_Buf[j])
    { // 低电平点亮段，段码[MSB...LSB]对应码顺序为[dp g f e d c b a]
      case '0': temp = 0xc0; break;
      case '1': temp = 0xf9; break;
      case '2': temp = 0xa4; break;
      case '3': temp = 0xb0; break;
      case '4': temp = 0x99; break;
      case '5': temp = 0x92; break;
      case '6': temp = 0x82; break;
      case '7': temp = 0xf8; break;
      case '8': temp = 0x80; break;
      case '9': temp = 0x90; break;

      case 'A': temp = 0x88; break;
      case 'B': temp = 0x83; break;
      case 'C': temp = 0xc6; break;
      case 'D': temp = 0xA1; break;
      case 'E': temp = 0x86; break;
      case 'F': temp = 0x8E; break;
      case 'H': temp = 0x89; break;
      case 'L': temp = 0xC7; break;
      case 'N': temp = 0xC8; break;
```



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
        case 'P': temp = 0x8c; break;
        case 'U': temp = 0xC1; break;
        case '-': temp = 0xbf; break;
        case ' ': temp = 0xff; break;
        default: temp = 0xff;
      }
      if(pucSeg_Buf[j+1] == '.')
      {
        temp = temp&0x7f;
        j++;
      }
      pucSeg_Code[i] = temp;
    }
  }
  // 数码管显示
  void Seg_Disp(unsigned char *pucSeg_Code, unsigned char ucSeg_Pos)
  {
    P0 = 0xff;                      // 消隐
    P2 = P2 & 0x1F | 0xE0;          // P27～P25 清零，再定位 Y7C
    P2 &= 0x1F;                     // P27～P25 清零
    P0 = 1<<ucSeg_Pos;              // 位选
    P2 = P2 & 0x1F | 0xC0;          // P27～P25 清零，再定位 Y6C
    P2 &= 0x1F;                     // P27～P25 清零
    P0 = pucSeg_Code[ucSeg_Pos];    // 段码
    P2 = P2 & 0x1F | 0xE0;          // P27～P25 清零，再定位 Y7C
    P2 &= 0x1F;                     // P27～P25 清零
  }
```

数码管示例程序“main.c”内容如下：

```
  // 运行程序时，将 J13 调整为 MM 模式（1、2 脚短接）
  #include "tim.h"
  #include "seg.h"
  #include "stdio.h"

  unsigned char ucSec, ucLed;
  unsigned char pucSeg_Buf[9], pucSeg_Code[8], ucSeg_Pos;
  unsigned long ulms;
  // 注意：sprintf()会在字符串后面添加”\0”，所以 pucSeg_Buf[]的长度应为 9。
  // 如果字符串中包含小数点，pucSeg_Buf[]的长度应为 10。
  void main(void)
  {
    Cls_Peripheral();
    Timer1Init();

    while(1)
    {
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
    sprintf(pucSeg_Buf, "   %04u", (unsigned int)ucSec);
    Seg_Tran(pucSeg_Buf, pucSeg_Code);
  }
}

void Time_1(void) interrupt 3
{
  ulms++;
  if(!(ulms % 1000))
  {
    ucSec++;

    ucLed ^= 1;
    Led_Disp(ucLed);
  }
  Seg_Disp(pucSeg_Code, ucSeg_Pos);
  if(++ucSeg_Pos == 8) ucSeg_Pos = 0;
}
```

数码管程序实现的方法如下：

（1）在“D:\CT107D”文件夹中将“203_TIM”文件夹复制并重命名为“204_SEG”文件夹，进入“204_SEG”文件夹，双击工程文件“IAP15”打开工程。

（2）在“Driver”文件夹中新建文件“seg.h”和“seg.c”。

（3）将“seg.c”添加到“Source Group 1”中。

（4）替换“main.c”的内容。

（5）编译并调试程序。

**注意**：运行程序时，将 J13 调整为 IO 模式（2、3 脚短接）。

## 3.5 矩阵键盘程序设计

矩阵键盘是单片机外部设备中所使用的排布类似于矩阵的键盘组。在键盘中按键数量较多时，为了减少 I/O 口的占用，通常将按键排列成矩阵形式。在矩阵键盘中，在每条水平线和垂直线交叉处不直接连通，而是通过一个按键加以连接。这样，一个端口（如 P0 口）就可以构成 4×4=16 个按键，比直接将端口线用于键盘多出了一倍，而且线数越多，区别越明显，比如再多加一条线就可以构成 20 键的键盘，而直接用端口线则只能多出一键（9 键）。由此可见，在需要的键数比较多时，采用矩阵键盘是合理的。

矩阵式结构的键盘显然比直接法要复杂一些，识别也要复杂一些，列线通过电阻接正电源，并将行线所接的单片机的 I/O 口作为输出端，而列线所接的 I/O 口则作为输入端。这样，当按键没有按下时，所有的输入端都是高电平，代表无键按下。行线输出是低电平，一旦有键按下，则输入线就会被拉低，这样，通过读入输入线的状态就可得知是否有键按下了。

矩阵键盘中按键的识别有行扫描和线翻转两种方法，其中行扫描法是一种最常用的按键识别方法，过程如下：

（1）逐行输出 0，检查列线是否非全高；

（2）如果某行输出 0 时，查到列线非全高，则该行有按键按下；



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

（3）根据输出 0 的行线和读入 0 的列线，即可判断按下按键的位置。
国信长天单片机竞赛实训平台上 4×4 矩阵键盘电路图如图 3.7 所示。

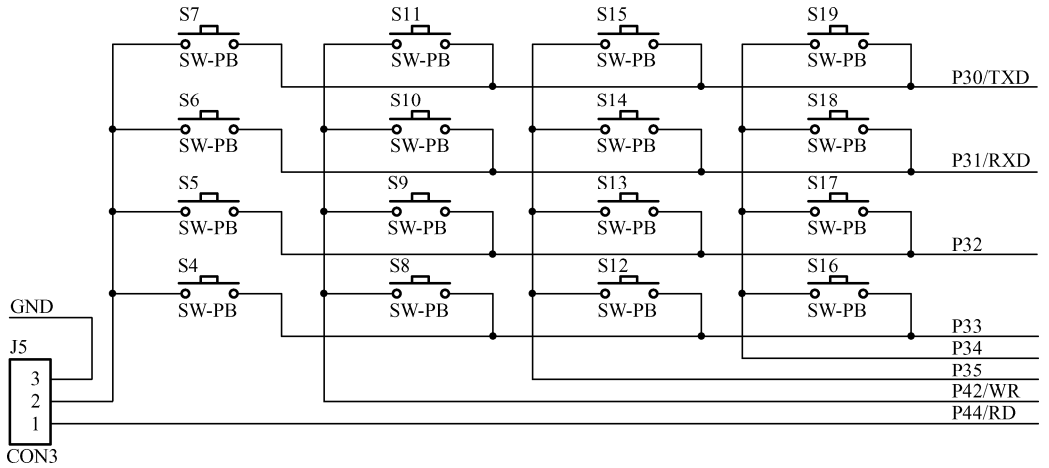



图 3.7　国信长天单片机竞赛实训平台上 4×4 矩阵键盘电路图

其中 J5 用于键盘选择，1、2 连接时为矩阵按键，2、3 连接时 S4～S7 为独立按键。
基于国信长天单片机竞赛实训平台的矩阵键盘程序设计如下：
矩阵键盘头文件“key.h”内容如下（存放在“D:\CT107D\Driver”文件夹中）：

```
#ifndef __KEY_H
#define __KEY_H
#include "STC15F2K60S2.H"

unsigned char Key_Read(void);
#endif
```

矩阵键盘源文件“key.c”内容如下（存放在“D:\CT107D\Driver”文件夹中）：

```
// 运行程序时，将 J5 调整为 KBD 模式（1、2 脚短接）
#include "key.h"

unsigned char Key_Read(void)
{
  unsigned int  Key_New;
  unsigned char Key_Val;

  P44 = 0; P42 = 1; P35 = 1; P34 = 1;    // 第 1 列
  Key_New = P3;
  P44 = 1; P42 = 0                       // 第 2 列
  Key_New = (Key_New<<4) | (P3&0x0f);
  P42 = 1; P35 = 0;                      // 第 3 列
  Key_New = (Key_New<<4) | (P3&0x0f);
  P35 = 1; P34 = 0;                      // 第 4 列
  Key_New = (Key_New<<4) | (P3&0x0f);

  switch(~Key_New)
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
  {
    case 0x8000: Key_Val = 4; break;     // S4
    case 0x4000: Key_Val = 5; break;     // S5
    case 0x2000: Key_Val = 6; break;     // S6
    case 0x1000: Key_Val = 7; break;     // S7
    case 0x0800: Key_Val = 8; break;     // S8
    case 0x0400: Key_Val = 9; break;     // S9
    case 0x0200: Key_Val = 10; break;    // S10
    case 0x0100: Key_Val = 11; break;    // S11
    case 0x0080: Key_Val = 12; break;    // S12
    case 0x0040: Key_Val = 13; break;    // S13
    case 0x0020: Key_Val = 14; break;    // S14
    case 0x0010: Key_Val = 15; break;    // S15
    case 0x0008: Key_Val = 16; break;    // S16
    case 0x0004: Key_Val = 17; break;    // S17
    case 0x0002: Key_Val = 18; break;    // S18
    case 0x0001: Key_Val = 19; break;    // S19
    default: Key_Val = 0;
  }
  return Key_Val;
}
```

矩阵键盘示例程序“main.c”内容如下:

```
#include "tim.h"
#include "key.h"
#include "seg.h"

unsigned char ucSec, ucLed;
unsigned char ucKey_Dly, ucKey_Old;
unsigned char pucSeg_Buf[8], pucSeg_Code[8], ucSeg_Pos;
unsigned long ulms, ulKey_Time;

void Key_Proc(void);
void SEG_Proc(unsigned char ucSeg_Val);

void main(void)
{
  Cls_Peripheral();
  Timer1Init();

  while(1)
  {
    Key_Proc();
    Seg_Tran(pucSeg_Buf, pucSeg_Code);
  }
}
```



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
void Time_1(void) interrupt 3
{
  ulms++;
  if(++ucKey_Dly == 10)
    ucKey_Dly = 0;
  if(!(ulms % 1000))
  {
    ucSec++;
    ucLed ^= 1;
    Led_Disp(ucLed);
  }
  Seg_Disp(pucSeg_Code, ucSeg_Pos);
  if(++ucSeg_Pos == 8) ucSeg_Pos = 0;
}

void Key_Proc(void)
{
  unsigned char ucKey_Val, ucKey_Down, ucKey_Up;

  if(ucKey_Dly) return;
  ucKey_Dly = 1;

  ucKey_Val = Key_Read();
  ucKey_Down = ucKey_Val & (ucKey_Old ^ ucKey_Val);
  ucKey_Up = ~ucKey_Val & (ucKey_Old ^ ucKey_Val);
  ucKey_Old = ucKey_Val;

  if(ucKey_Down)
  {
    ulKey_Time = ulms;
    SEG_Proc(ucKey_Down);
  }
  if(ucKey_Up)
    SEG_Proc(ucKey_Up);
  if(ucKey_Old && (ulms - ulKey_Time > 1000))
  {
    ulKey_Time = ulms;
    SEG_Proc(ucKey_Old);
  }
}

void SEG_Proc(unsigned char ucSeg_Val)
{
  unsigned char i;
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
    for(i=0; i<7; i++)
      pucSeg_Buf[i] = pucSeg_Buf[i+1];
    if(ucSeg_Val < 14)
      pucSeg_Buf[i] = ucSeg_Val-4+'0';
    else
      pucSeg_Buf[i] = ucSeg_Val-14+'A';
  }
```

矩阵键盘程序实现的方法如下。

（1）在“D:\CT107D”文件夹中将“204_SEG”文件夹复制并重命名为“205_KEY”文件夹，进入“205_KEY”文件夹，双击工程文件“IAP15”打开工程。

（2）在“Driver”文件夹中新建文件“key.h”和“key.c”。

（3）将“key.c”添加到“Source Group 1”中。

（4）替换“main.c”的内容。

（5）编译并调试程序，按下或松开按键，对应的键值显示在数码管上，长按按键，每 1s 显示一次对应的键值。

**注意：**运行程序时，将 J5 调整为 KBD 模式（1、2 脚短接），将 J13 调整为 IO 模式（2、3 脚短接）。因为 MM 模式使用了 P42，与按键 KBD 模式冲突，所以 J5 使用 KBD 模式时 J13 不能使用 MM 模式。

## 3.6 串口程序设计

IAP15F2K61S2 单片机内部有 2 个采用 UART 工作方式的全双工串行通信接口（串行口 1 和串行口 2）。每个串行口由 2 个数据缓冲器、1 个移位寄存器、1 个串行控制器和 1 个波特率发生器等组成。每个串行口的数据缓冲器由 2 个相互独立的接收、发送缓冲器构成，可以同时发送和接收数据。发送数据缓冲器只能写入而不能读出，接收缓冲器只能读出而不能写入，因而两个缓冲器可以共用一个地址码。串行口 1 的两个数据缓冲器的共用地址码是 99H，串行口 2 的两个数据缓冲器的共用地址码是 9BH。

IAP15F2K61S2 单片机的串行口有 4 种工作方式，有的工作方式的波特率是可变的。用户用软件编程的方法在串行口控制寄存器中写入相应的控制字节，即可改变串行口的波特率和工作方式。

IAP15F2K61S2 单片机串行口 1 对应的硬件部分是 TxD 和 RxD，串行口 2 对应的硬件部分是 TxD2 和 RxD2，通过设置特殊功能寄存器 AUXR1 和 S1_S1 和 S1～S0，串行口 1（UART1）可以在[P30/RxD，P31/TxD]、[P36/RxD_2，P37/TxD_2]、[P16/RxD_3，P17/TxD_3]之间切换。通过设置特殊功能寄存器 P_SW2 中的 S2_S 位，串行口 2（UART2）的功能可以在 P1 口和 P4 口之间任意切换。当串行口 2 功能在 P1 口实现时，对应的引脚是 P10/RxD2 和 P11/TxD2；当串行口 2 的功能在 P4 口实现时，对应的引脚是 P46/RxD2_2 和 P47/TxD2_2。

与串行口 1 相关的寄存器有：SCON、PCON、AUXR、SBUF、TMOD、TL1、TH1、TCON、IE、IP、CLK_DIV、P_SW2、SADEN 和 SADDR。与串行口 2 相关的寄存器有：S2CON、S2BUF、T2H、T2L、AUXR、IE2、IP2 和 AUXR1。

其中寄存器 TMOD、TL1、TH1 和 TCON 与定时器有关；寄存器 IE、IE2、IP 和 IP2 与中断有关；寄存器 P_SW2 和 AUXR1 与串行口的引脚切换有关。

基于国信长天单片机竞赛实训平台的串口程序设计如下：



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

串口头文件“uart.h”内容如下：

```
#ifndef __UART_H
#define __UART_H
#include "STC15F2K60S2.H"

void UartInit(void);
void Uart_SendString(unsigned char *pucStr);
#endif
```

串口源文件“uart.c”内容如下：

```
#include "uart.h"

void UartInit(void)                    //4800bps@12.000MHz
{
  SCON = 0x50;                         // 8 位数据，可变波特率
  AUXR |= 0x01;                        // 串口 1 选择定时器 2 为波特率发生器
  AUXR |= 0x04;                        // 定时器 2 时钟为 Fosc，即 1T
  T2L = 0x8F;                          // 设定定时初值
  T2H = 0xFD;                          // 设定定时初值
  AUXR |= 0x10;                        // 启动定时器 2
  ES = 1;                              // 允许串口中断
}

void Uart_Sendstring(unsigned char *pucStr)
{
  while(*pucStr != '\0')
  {
    SBUF = *pucStr;
    while(TI == 0);
    TI = 0;
    pucStr++;
  }
}
```

串口示例程序“main.c”内容如下：

```
#include "tim.h"
#include "seg.h"
#include "stdio.h"
#include "uart.h"

unsigned char ucSec, ucSec1, ucLed;
unsigned char pucSeg_Buf[11], pucSeg_Code[8], ucSeg_Pos;
unsigned char pucUart_Buf[2], ucUart_Num;
unsigned long ulms;
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
void Uart_Proc(void);

void main(void)
{
  Cls_Peripheral();
  Timer1Init();
  UartInit();

  while(1)
  {
    sprintf(pucSeg_Buf, "%08u\r\n", (unsigned int)ucSec);
    Seg_Tran(pucSeg_Buf, pucSeg_Code);
    Uart_Proc();
  }
}

void Time_1(void) interrupt 3
{
  ulms++;
  if(!(ulms % 1000))
  {
    ulms = 0;
    ucSec++;

    ucLed ^= 1;
    Led_Disp(ucLed);
  }
  Seg_Disp(pucSeg_Code, ucSeg_Pos);
  if(++ucSeg_Pos == 8) ucSeg_Pos = 0;
}

void Uart_Proc(void)
{
  if(ucSec1 != ucSec)
  {
    ucSec1 = ucSec;
    Uart_SendString(pucSeg_Buf);
  }
  if(ucUart_Num == 2)
  {
    ucSec = ((pucUart_Buf[0]-'0')*10) + pucUart_Buf[1]-'0';
    ucUart_Num = 0;
  }
}

void uart_0(void) interrupt 4
```



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
{
  if(RI)
  {
    pucUart_Buf[ucUart_Num++] = SBUF;
    RI = 0;
  }
}
```

串口程序实现的方法如下：

（1）在“D:\CT107D”文件夹中将“204_SEG”文件夹复制并重命名为“206_UART”文件夹，进入“206_UART”文件夹，双击工程文件“IAP15”打开工程。

（2）在“Driver”文件夹中新建文件“uart.h”和“uart.c”。

（3）将“uart.c”添加到“Source Group 1”中。

（4）替换“main.c”的内容。

（5）编译并用 STC-ISP 将 IAP15.hex 下载到开发板运行。（选择 IRC 的频率为 12.000MHz，否则串口助手中会出现错误结果。）

（6）用 STC-ISP 串口助手可以接收开发板发送的字符串，并向开发板发送字符串，如图 3.8 所示。

图 3.8　串口助手

**注意**：由于下载调试使用的也是串口，所以串口程序无法使用 Keil 调试运行，可以在其他程序调试通过后再添加串口程序下载运行。同时下载运行程序后，为了在后续的使用中能够调试程序，必须使用 STC-ISP 的 Keil 仿真设置重新将 IAP15F2K61S2/IAP15L2K61S2 设置为仿真芯片。

## 3.7　DS1302 时钟芯片驱动程序设计

时钟芯片 DS1302 是 DALLAS 公司推出的涓流充电时钟芯片，内含一个实时时钟/日历和 31 字节静态 RAM，通过简单的串行接口与单片机进行通信。实时时钟/日历电路提供秒、分、时、日、周、月、年的信息，每月的天数和闰年的天数可自动调整。

时钟操作可通过 AM/PM 指示决定采用 24 或 12 小时格式。DS1302 与单片机之间能简单地采用同步串行方式进行通信，仅用到三个口线：RES 复位、I/O 数据和 SCLK 串行时钟。时钟

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

/RAM 的读/写数据以一个字节或多达 31 个字节的字符组方式通信。

DS1302 有 12 个寄存器，其中有 7 个寄存器与日历、时钟相关，存放的数据位为 BCD 码形式，即日历、时间寄存器。此外，DS1302 还有年份寄存器、控制寄存器、充电寄存器、时钟突发寄存器及与 RAM 相关的寄存器等。时钟突发寄存器可一次性顺序读写除充电寄存器外的所有寄存器内容。

DS1302 与 RAM 相关的寄存器分为两类：一类是单个 RAM 单元，共 31 个，每个单元组态为一个 8 位的字节，其命令控制字为 C0H-FDH，其中奇数为读操作，偶数为写操作；另一类为突发方式下的 RAM 寄存器，此方式下可一次性读写所有的 RAM 的 31 个字节，命令控制字为 FEH（写）、FFH（读）。

操作 DS1302 的大致过程，就是将各种数据写入 DS1302 的寄存器，以设置它当前的时间格式。然后使 DS1302 开始运作，DS1302 时钟会按照设置情况运转，再用单片机将其寄存器内的数据读出。

DS1302 的控制字节如图 3.9 所示。

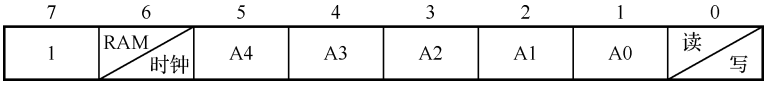



图 3.9　DS1302 控制字节

DS1302 写字节时序如图 3.10 所示。第一个字节是地址字节，第二个字节是数据字节。RST 信号必须拉高，否则数据的输入是无效的。换句话说，RST 信号控制数据时间信号输入的开始和结束。当地址字节和数据字节读取时上升沿有效，而且由 LSB 开始读入。

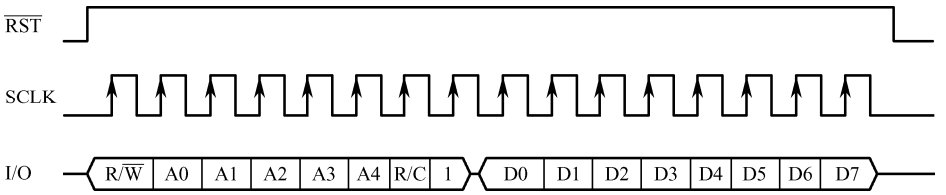



图 3.10　DS1302 写字节时序

DS1302 读字节时序如图 3.11 所示。读一个字节和写一个字节有明显的不同，先是写地址字节，然后再读数据字节，写地址字节时上升沿有效，而读数据字节时下降沿有效，当然前提是 RST 信号必须拉高。写地址字节和读数据字节同是从最低位（即第 0 位）开始的。

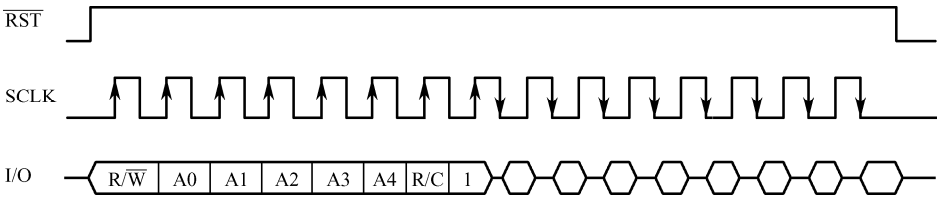



图 3.11　DS1302 读字节时序

DS1302 的时钟寄存器如表 3.7 所示。

**表 3.7　DS1302 时钟寄存器**

| 地址 | 数据 | | | | | 初始值 | 说明 |
|---|---|---|---|---|---|---|---|
| 0 | CH | 秒十位 | | | 秒个位 | 0x80 | 秒：00～59，CH=1：时钟暂停 |
| 1 | 0 | 分十位 | | | 分个位 | 0 | 分：00～59 |
| 2 | 24/12 | 0 | 时 A/P | 时 | 时个位 | 0 | 时：00～23/01～12 |



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

续表

| 地址 | 数据 | | | | | | | | 初始值 | 说明 |
|---|---|---|---|---|---|---|---|---|---|---|
| 3 | 0 | 0 | 日十位 | | 日个位 | | | | 1 | 日：01～28/29/30/31 |
| 4 | 0 | 0 | 0 | 月 | 月个位 | | | | 1 | 月：01～12 |
| 5 | 0 | 0 | 0 | 0 | 0 | 星期 | | | 1 | 星期：1～7 |
| 6 | 年十位 | | | | 年个位 | | | | 0 | 年：00～99 |
| 7 | WP | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 控制，WP=1：写保护 |
| 8 | TCS | | | | DS | | RS | | 0x5c | 涓流充电选择 |
| 31 | | | | | | | | | - | 时钟突发 |

国信长天单片机竞赛实训平台的 DS1302 电路图如图 3.12 所示。

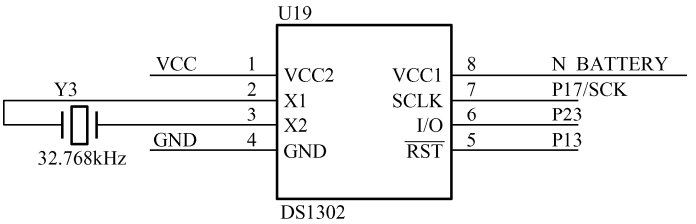



图 3.12　国信长天单片机竞赛实训平台 DS1302 电路图

竞赛提供的资源包中包含 DS1302 的头文件 ds1302.h 和源文件 ds1302.c，其中包含下列 3 个函数：

```
void Write_Ds1302(unsigned char temp);
void Write_Ds1302_Byte(unsigned char addr, unsigned char dat);
unsigned char Read_Ds1302_Byte(unsigned char addr);
```

利用这 3 个函数，并参考图 3.9 和表 3.7，可以编写实时钟操作函数如下：

在“ds1302.h”中追加下列声明：

```
void Set_RTC(unsigned char* pucRtc);
void Read_RTC(unsigned char* pucRtc);
```

在“ds1302.c”中追加下列源代码：

```
// 设置时钟
void Set_RTC(unsigned char* pucRtc)
{
  unsigned char temp;

  Write_Ds1302_Byte(0x8E, 0);                    // WP=0：允许写操作
  temp = ((pucRtc[0]/10)<<4)+pucRtc[0]%10;
  Write_Ds1302_Byte(0x84, temp);                 // 设置时
  temp = ((pucRtc[1]/10)<<4)+pucRtc[1]%10;
  Write_Ds1302_Byte(0x82, temp);                 // 设置分
  temp = ((pucRtc[2]/10)<<4)+pucRtc[2]%10;
  Write_Ds1302_Byte(0x80, temp);                 // 设置秒
  Write_Ds1302_Byte(0x8E, 0x80);                 // WP=1：禁止写操作
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
}
// 读取时钟
void Read_RTC(unsigned char* pucRtc)
{
  unsigned char temp;

  temp = Read_Ds1302_Byte(0x85);                    // 读取时
  pucRtc[0] = (temp>>4)*10+(temp&0xf);
  temp = Read_Ds1302_Byte(0x83);                    // 读取分
  pucRtc[1] = (temp>>4)*10+(temp&0xf);
  temp = Read_Ds1302_Byte(0x81);                    // 读取秒
  pucRtc[2] = (temp>>4)*10+(temp&0xf);
}
```

基于国信长天单片机竞赛实训平台的时钟芯片示例程序“main.c”内容如下：

```
#include "tim.h"
#include "key.h"
#include "seg.h"
#include "stdio.h"
#include "ds1302.h"

unsigned int  uiSeg_Dly;
unsigned char ucState=1, ucSec, ucLed;
unsigned char ucKey_Dly, ucKey_Old;
unsigned char pucSeg_Buf[9], pucSeg_Code[8], ucSeg_Pos;
unsigned char pucRtc[3] = {23, 59, 50};
unsigned long ulms;

void Key_Proc(void);
void Seg_Proc(void);

void main(void)
{
  Cls_Peripheral();
  Timer1Init();
  Set_RTC(pucRtc);

  while(1)
  {
    Key_Proc();
    Seg_Proc();
  }
}

void Time_1(void) interrupt 3
{
```



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
  ulms++;
  if(++ucKey_Dly == 10)
    ucKey_Dly = 0;
  if(++uiSeg_Dly == 500)
    uiSeg_Dly = 0;
  if(!(ulms % 1000))
    ucSec++;
  Seg_Disp(pucSeg_Code, ucSeg_Pos);
  if(++ucSeg_Pos == 8) ucSeg_Pos = 0;
}

void Key_Proc(void)
{
  unsigned char ucKey_Val, ucKey_Down;

  if(ucKey_Dly) return;
  ucKey_Dly = 1;

  ucKey_Val = Key_Read();
  ucKey_Down = ucKey_Val & (ucKey_Old ^ ucKey_Val);
  ucKey_Old = ucKey_Val;

  if(ucKey_Down == 4)
    if(++ucState == 2)
      ucState = 0;

  Led_Disp(ucState+1);
}

void Seg_Proc(void)
{
  if(ucKey_Dly) return;
  ucKey_Dly = 1;

  switch(ucState)
  {
    case 0:
      sprintf(pucSeg_Buf, "1 %06u", (unsigned int)ucSec);
      break;
    case 1:
      Read_RTC(pucRtc);
      sprintf(pucSeg_Buf, "2 %02d%02d%02d", (unsigned int)pucRtc[0],\
        (unsigned int)pucRtc[1], (unsigned int)pucRtc[2]);
  }
  Seg_Tran(pucSeg_Buf, pucSeg_Code);
}
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

实时钟程序实现的方法如下。

（1）在“D:\CT107D”文件夹中将“205_KEY”文件夹复制并重命名为“207_DS1302”文件夹，进入“207_DS1302”文件夹，双击工程文件“IAP15”打开工程。

（2）将竞赛资源包中的“ds1302.h”和“ds1302.c”复制粘贴到“Driver”文件夹中。

（3）分别在“ds1302.h”“ds1302.c”中追加 Set_RTC()和 Read_RTC()的声明及源代码。

（4）将“ds1302.c”添加到“Source Group 1”中。

（5）替换“main.c”的内容。

（6）编译并调试程序，按下按键 S4，数码管分别显示定时值或实时钟。

## 3.8 DS18B20 温度传感器驱动程序设计

DS18B20 是单线接口数字温度传感器，测量范围是−55℃～+125℃，−10℃～+85℃范围内精度是±0.5℃，测量分辨率为 9～12 位（复位值为 12 位，最大转换时间为 750ms）。

DS18B20 包括寄生电源电路、64 位 ROM 和单线接口电路、暂存器、EEPROM、8 位 CRC 生成器和温度传感器等。寄生电源电路可以实现外部电源供电和单线寄生供电，64 位 ROM 中存放的 48 位序列号用于识别同一单线上连接的多个 DS18B20，以实现多点测温。

64 位 ROM 代码的格式为：

8 位 CRC 校验码 + 48 位序列号 + 8 位系列码（0x28）

其中 8 位 CRC 校验码是 48 位序列号和 8 位系列码的 CRC 校验码。

DS18B20 的暂存器如表 3.8 所示。

表 3.8 DS18B20 暂存器

| 地　址 | 名　称 | 类　型 | 复 位 值 | 说　明 |
|---|---|---|---|---|
| 0 | 温度值低 8 位 | 只读 | 0x0550 | b15～b11：符号位，b10～b4：7 位整数 |
| 1 | 温度值高 8 位 | 只读 | （85℃） | b3～b0：4 位小数（补码） |
| 2 | TH 或用户字节 1 | 读写 | EEPROM | b7：符号位，b6～b0：7 位温度报警高值（补码） |
| 3 | TL 或用户字节 2 | 读写 | EEPROM | b7：符号位，b6～b0：7 位温度报警低值（补码） |
| 4 | 配置寄存器 CR | 读写 | EEPROM | b6～b5：分辨率，00～11：9～12 位 |
| 5:7 | 保留 | 只读 | 0x100CFF | |
| 8 | CRC | 只读 | EEPROM | 暂存器 0～7 数据 CRC 校验码 |

DS18B20 的操作包括下列 3 步：

- 复位
- ROM 命令
- 功能命令

DS18B20 ROM 命令和功能命令分别如表 3.9 和表 3.10 所示。

表 3.9 DS18B20 ROM 命令

| 命　令 | 代　码 | 参数或返回值 | 说　明 |
|---|---|---|---|
| 搜索 ROM | 0xF0 | - | 搜索单线上连接的多个 DS18B20，搜索后重新初始化 |
| 读取 ROM | 0x33 | ROM 代码 | 读取单个 DS18B20 的 64 位 ROM 代码 |
| 匹配 ROM | 0x55 | ROM 代码 | 寻址指定 ROM 代码的 DS18B20 |
| 跳过 ROM | 0xCC | - | 寻址所有单线上连接的多个 DS18B20 |
| 搜索报警 | 0xEC | | 搜索单线上连接的有报警标志的 DS18B20 |



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

表 3.10　DS18B20 功能命令

| 命　令 | 代　码 | 参数或返回值 | 说　明 |
|---|---|---|---|
| 转换温度 | 0x44 | 0-转换，1-完成 | 启动温度转换，转换结果存放在暂存器的 0～1 字节 |
| 读暂存器 | 0xBE | 9 字节数据 | 读取暂存器的 0～8 字节 |
| 写暂存器 | 0x4E | TH TL CR | 将 TH、TL 和 CR 值写入暂存器的 2～4 字节 |
| 复制暂存器 | 0x48 | - | 将暂存器的 2～4 字节复制到 EEPROM |
| 调回 EEPROM | 0xB8 | 0-调回，1-完成 | 将 EEPROM 的值调回到暂存器的 2～4 字节 |
| 读电源模式 | 0xB4 | - | 确定 DS18B20 是否使用寄生供电模式 |

国信长天单片机竞赛实训平台的 DS18B20 电路图如图 3.13 所示。

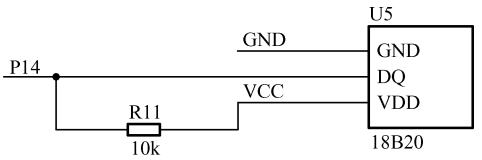



图 3.13　国信长天单片机竞赛实训平台 DS18B20 电路图

竞赛提供的资源包中包含 DS18B20 的头文件 onewire.h 和源文件 onewire.c，其中源文件包含下列 3 个函数：

```
void Write_DS18B20(unsigned char dat);
unsigned char Read_DS18B20(void);
bit init_ds18b20(void);                    // 返回值为存在信号：0-存在
```

利用这 3 个函数，并参考表 3.8～表 3.10，可以编写读取温度函数如下：

将“onewire.h”中的下列声明：

```
unsigned char rd_temperature(void);
```

修改为：

```
unsigned int rd_temperature(void);
```

在“onewire.c”中追加下列源代码：

```
unsigned int rd_temperature(void)
{
  unsigned char low, high;

  init_ds18b20();                          // 初始化
  Write_DS18B20(0xCC);                     // 跳过 ROM
  Write_DS18B20(0x44);                     // 转换温度

  init_ds18b20();
  Write_DS18B20(0xCC);
  Write_DS18B20(0xBE);                     // 读暂存器
  low = Read_DS18B20();                    // 低字节
  high = Read_DS18B20();                   // 高字节
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
  return (high<<8)+low;
}
```

基于国信长天单片机竞赛实训平台的温度传感器示例程序“main.c”内容如下：

```
#include "tim.h"
#include "key.h"
#include "seg.h"
#include "stdio.h"
#include "onewire.h"

unsigned int  uiSeg_Dly, uiTemp;
unsigned char ucState=1, ucSec, ucLed;
unsigned char ucKey_Dly, ucKey_Old;
unsigned char pucSeg_Buf[9], pucSeg_Code[8], ucSeg_Pos;
unsigned long ulms;

void Key_Proc(void);
void Seg_Proc(void);

void main(void)
{
  Cls_Peripheral();
  Timer1Init();

  while(1)
  {
    Key_Proc();
    Seg_Proc();
  }
}

void Time_1(void) interrupt 3
{
  ulms++;
  if(++ucKey_Dly == 10)
    ucKey_Dly = 0;
  if(++uiSeg_Dly == 500)
    uiSeg_Dly = 0;
  if(!(ulms % 1000))
    ucSec++;
  Seg_Disp(pucSeg_Code, ucSeg_Pos);
  if(++ucSeg_Pos == 8) ucSeg_Pos = 0;
}

void Key_Proc(void)
{
```



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
  unsigned char ucKey_Val, ucKey_Down;

  if(ucKey_Dly) return;
  ucKey_Dly = 1;

  ucKey_Val = Key_Read();
  ucKey_Down = ucKey_Val & (ucKey_Old ^ ucKey_Val);
  ucKey_Old = ucKey_Val;

  if(ucKey_Down == 4)
    if(++ucState == 2)
      ucState = 0;

  Led_Disp(ucState+1);
}

void Seg_Proc(void)
{
  if(uiSeg_Dly) return;
  uiSeg_Dly = 1;

  switch(ucState)
  {
    case 0:
      sprintf(pucSeg_Buf, "1 %06u", (unsigned int)ucSec);
      break;
    case 1:
      uiTemp = rd_temperature();
      sprintf(pucSeg_Buf, "2   %04.2f", uiTemp/16.0);
  }
  Seg_Tran(pucSeg_Buf, pucSeg_Code);
}
```

温度传感器程序实现的方法如下。

（1）在“D:\CT107D”文件夹中将“205_KEY”文件夹复制并重命名为“208_DS18B20”文件夹，进入“208_DS18B20”文件夹，双击工程文件“IAP15”打开工程。

（2）将竞赛资源包中的“onewire.h”和“onewire.c”复制粘贴到“Driver”文件夹中。

（3）修改“onewire.h”内容，将上述 rd_temperature()源代码追加到“onewire.c”中。

（4）将“onewire.c”添加到“Source Group 1”中。

（5）修改“main.c”的内容。

（6）编译并调试程序，按下按键 S4，数码管分别显示定时值或温度值，用手捏住温度传感器 U5，温度值增大，放开后温度值减小。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

## 3.9 PCF8591 ADC/DAC 驱动程序设计

在一个完整的单片机系统中，A/D 转换芯片往往是必不可少的。在与 CPU 的信息传输过程中仅靠时钟线 SCL 和数据线 SDA 就可以实现。PCF8591 是具有 I2C 总线接口的 8 位 A/D 及 D/A 转换器，有 4 路 A/D 转换输入，1 路 D/A 模拟输出。

PCF8591 采用典型的 I2C 总线接口器件寻址方法，即总线地址由器件地址、引脚地址和方向位组成。飞利浦公司规定 A/D 器件地址为 1001。引脚地址为 A2、A1、A0，其值由用户选择，因此 I2C 系统中最多可接 8 个具有 I2C 总线接口的 A/D 器件。地址的最后一位为方向位 R/W，当主控器对 A/D 器件进行读操作时为 1，进行写操作时为 0。总线操作时，由器件地址、引脚地址和方向位组成的从地址为主控器发送的第一字节。

控制字节用于实现器件的各种功能，如模拟信号由哪几个通道输入等。控制字节存放在控制寄存器中，总线操作时为主控器发送的第二字节，其格式如图 3.14 所示。

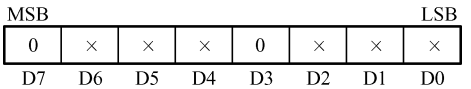



图 3.14　控制字节格式

其中，D1 和 D0 是 A/D 通道编号：00——通道 0，01——通道 1，10——通道 2，11——通道 3；

D2 是自动增益选择（1 有效）；

D5 和 D4 是模拟量输入选择：00——四路单端输入、01——三路差分输入、10——两两单端与一路差分、11——两路差分；

D6 是模拟输出允许（1 有效）。

当系统为 A/D 转换时，模拟输出允许为 0。模拟量输入选择位取值由输入方式决定：四路单端输入时取 00，三路差分输入时取 01，单端与差分输入时取 10，二路差分输入时取 11。最低两位是通道编号位，当对 0 通道的模拟信号进行 A/D 转换时取 00，当对 1 通道的模拟信号进行 A/D 转换时取 01，当对 2 通道的模拟信号进行 A/D 转换时取 10，当对 3 通道的模拟信号进行 A/D 转换时取 11。

在进行数据操作时，首先是主控器发出起始信号，然后发出读寻址字节，被控器发出应答后，主控器从被控器读出第一个数据字节，主控器发出应答，主控器从被控器读出第二个数据字节，主控器发出应答，一直到主控器从被控器中读出第 *n* 个数据字节，主控器发出非应答信号，最后主控器发出停止信号。

发送给 PCF8591 的第三个字节被存储到 DAC 数据寄存器，并使用片上 D/A 转换器转换成对应的模拟电压。这个 D/A 转换器由连接至外部参考电压的具有 256 个接头的电阻分压电路和选择开关组成。

国信长天单片机竞赛实训平台的 PCF8591 电路图如图 3.15 所示。将器件地址输入端 A0～A2 接地；I2C 总线的数据线 SCL 和时钟线 SDA 分别和单片机引脚 P20 和 P21 相连；内部时钟 EXT 接地。



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

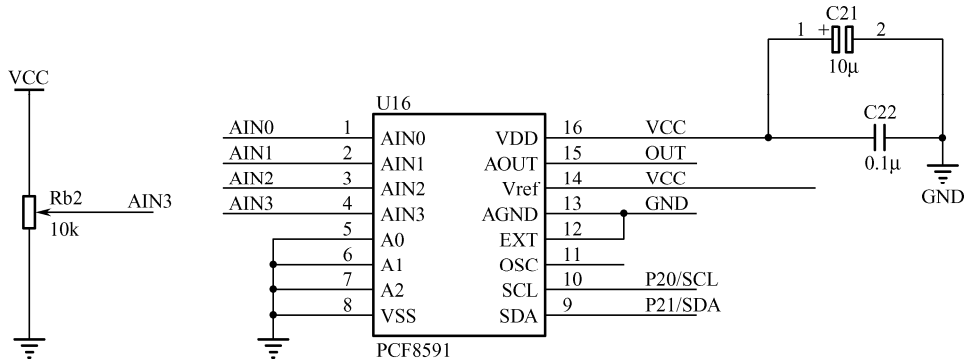



图 3.15　国信长天单片机竞赛实训平台 PCF8591 电路图

竞赛资源包中包含 I2C 的头文件“iic.h”和源文件“iic.c”，其中包含下列 6 个函数：

```
void IIC_Start(void);
void IIC_Stop(void);
bit IIC_WaitAck(void);
void IIC_SendAck(bit ackbit);
void IIC_SendByte(unsigned char byt);
unsigned char IIC_RecByte(void);
```

利用这 6 个函数，可以编写 PCF8591 的读写程序如下：
在“iic.h”中添加下列声明：

```
unsigned char PCF8591_Adc(void);
void PCF8591_Dac(unsigned char dat);
```

在“iic.c”中添加下列源代码：

```
#ifdef PCF8591_ADC
// 通过 I2C 总线读取 ADC 结果
unsigned char PCF8591_Adc(void)
{
  unsigned char temp;

  IIC_Start();
  IIC_SendByte(0x90);
  IIC_WaitAck();

  IIC_SendByte(0x43);              // 允许 DAC，ADC 通道 3
  IIC_WaitAck();

  IIC_Start();
  IIC_SendByte(0x91);
  IIC_WaitAck();

  temp = IIC_RecByte();
  IIC_SendAck(1);
  IIC_Stop();
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
  return temp;
}
#endif
#ifdef PCF8591_DAC
// DAC 输出
void PCF8591_Dac(unsigned char dat)
{
  IIC_Start();
  IIC_SendByte(0x90);
  IIC_WaitAck();

  IIC_SendByte(0x43);                  // 允许 DAC，ADC 通道 3
  IIC_WaitAck();

  IIC_SendByte(dat);                   // dat-输出数模转换的数据
  IIC_WaitAck();
  IIC_Stop();
}
#endif
```

PCF8591 读写程序的使用方法如下。

（1）将竞赛资源包中的“iic.h”和“iic.c”复制并重命名到“D:\CT107D\Driver”文件夹中。

（2）在“iic.h”“iic.c”中添加 PCF8591_adc()和 PCF8591_dac()的声明及源代码。

（3）将“iic.c”添加到“Source Group 1”中。

（4）将“iic.h”包含在“main.h”中。

（5）在 while(1)中调用 PCF8591_adc()或 PCF8591_dac()。

（6）在“Options for Target”的“C51”标签中定义预处理符号“PCF8591_ADC”或“PCF8591_DAC”，如图 3.16 所示。

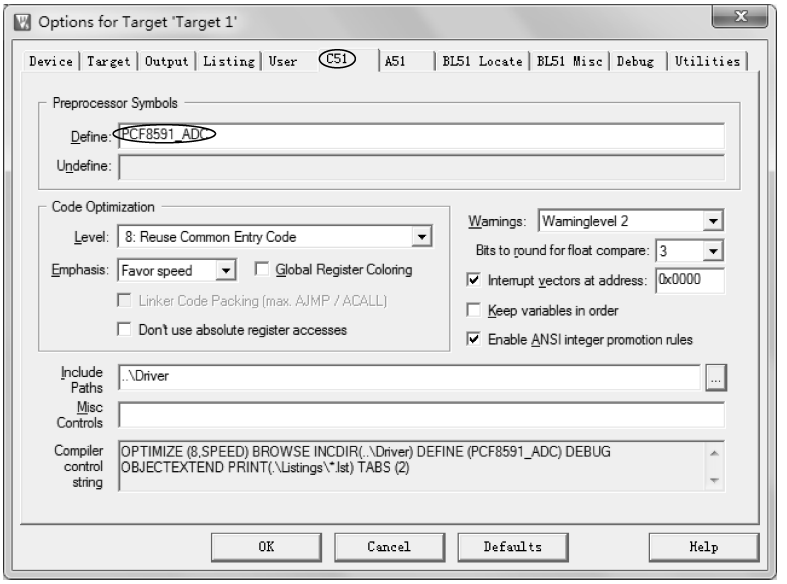



图 3.16　预处理符号定义

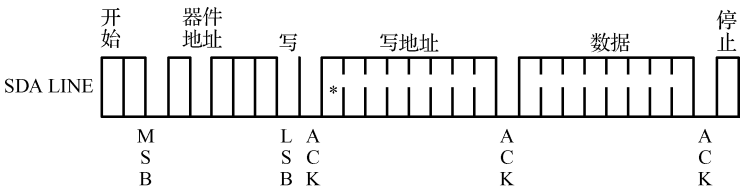


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

## 3.10　AT24C02 存储器驱动程序设计

AT24C02 存储器为低功耗 CMOS 工艺制造，可单电压工作。AT24C02 存储器特点如下：容量为 256×8（2KB）；双线串行接口（双线为：时钟线 SCL，串行数据线 SDA）；双线数据传送协议；支持 ISO/IEC7816-3 同步协议；8 字节页面（1KB，2KB）、16 字节页面（4KB，8KB，16KB）写入方式；允许部分页面写入；自定时写入周期（最大 10ms）；高可靠性，擦写次数可达 10 万个周期，数据保存期达 100 年；以晶片、模块及标准封装等形式提供。标准封装有 8 个引脚的双列直插（PDIP）、8 个引脚的表面封装和 14 个引脚的表面封装（SOIC）。

AT24C02 存储器写操作需要在给出开始态、器件地址和确认之后，紧跟着给出一个 8 位数据地址。一经收到该地址，EEPROM 就通过 SDA 发出确认信号，并随时钟输入 8 位数据。在收到 8 位数据之后，EEPROM 将向 SDA 确认，数据传送设备必须用停止状态来终止写操作，这时，EEPROM 进入一个内计时固定存储器写入周期。在该写周期时，所有输入被禁止，EEPROM 直到写完后才应答，写字节时序如图 3.17 所示。




（ACK：确认　MSB：最高位　LSB：最低位　*：对1KB芯片无用）

图 3.17　写字节时序

1KB/2KB EEPROM 能进行 8 字节页面写入，4KB、8KB 和 16KB 设备能进行 16 字节页面写入。激发写页面与激发写字节相同，只是数据传送设备无须在第一个字节随时钟输入之后，发出一个停止状态。在 EEPROM 确认收到第一个数据之后，数据传送设备能再传送 7 个（1KB、2KB）或 15 个（4KB、8KB、16KB）数据，每一个数据收到之后，EEPROM 都将通过 SDA 回送一个确认信号，最后数据传送设备必须通过停止状态终止页面写序列。写页面时序如图 3.18 所示。

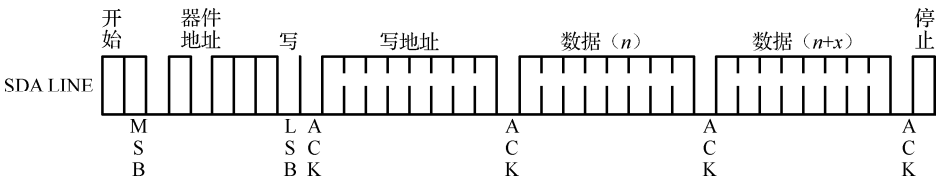



图 3.18　写页面时序

数据字地址的低 3 位（1KB、2KB）或低 4 位（4KB、8KB、16KB）在收到每个数据字后，内部自动加 1，数据字地址的高位字节保持不变，以保持存储器页地址不变。如果传送到 EEPROM 中的数据字超过 8（1KB、2KB）或 16（4KB、8KB、16KB）字节，数据字地址将重复滚动，以前的数据将被覆盖。

一旦内计时固定存储器写入周期开始，EEPROM 输入就被禁止，只有在内写循环完成时，EEPROM 才通过 SDA 总线上的确认应答，允许读或写过程继续进行。

除了器件地址码中读/写选择位置 1，读操作与写操作是一样的。读操作有三种：立即地址读取、随机地址读取和顺序读取。

立即地址读取时序如图 3.19 所示。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

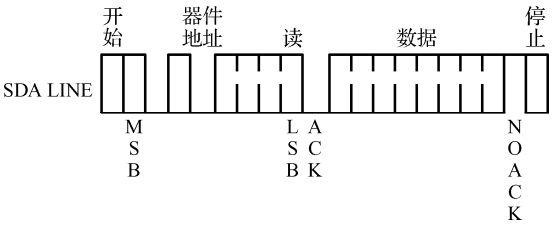



图 3.19　立即地址读取时序

内部数据字地址指针保持在读写操作中最后访问的地址，按“1”递增。只要芯片保持上电，该地址在两个操作之间一直有效，如果最后一个操作是在地址 *n* 处读取，则立即地址是 *n*+1；如果最后操作是在地址 *n* 处写入，则立即地址也是 *n*+1。

有一种情况例外，如果地址 *n* 是存储列中的第 8 个（1KB、2KB）或第 16 个（4KB、8KB、16KB）字节地址，则增加的地址 *n*+1 将回到同一列的第一个字节地址。一旦读/写选择位置 1，器件地址随时钟输入，并被 EEPROM 确认，立即寻址数据就随时钟串行输出。读数据的器件不是通过确认（使总线处于高电平）来应答，而是随后产生一个停止状态。

随机地址读取时序如图 3.20 所示。随机读取需要一个空字节写序列来载入数据地址，一旦器件地址码和数据码地址码时钟输入，并被 EEPROM 确认，数据传送设备就必须产生另一个开始条件。读/写选择位处于高电平时，通过送出一个器件地址，数据传送设备激发出一个立即寻址读取，EEPROM 确认器件地址，并随时钟串行输出数据。器件读数据不通过确认（使 SDA 总线处于高电平）应答，而通过产生一个停止条件应答。

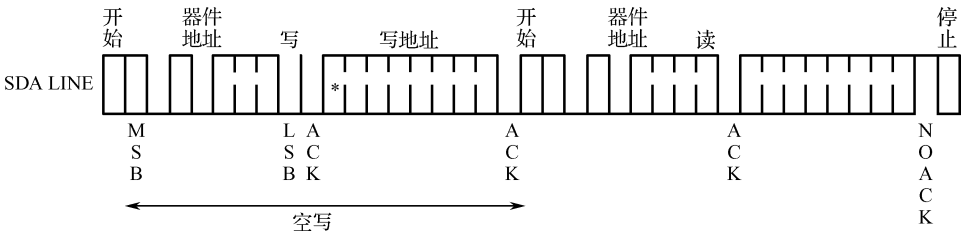



图 3.20　随机地址读取时序

顺序读取时序如图 3.21 所示。顺序读取由立即地址读取或随机地址读取激发，在读数据器件收到一数据码之后，通过确认应答，只要 EEPROM 收到确认之后，便会继续增加数据码地址及串行输出数据码。当达到存储器地址极限时，数据码地址将重复滚动，顺序读取将继续。当读数据器件不通过确认（使 SDA 总线处于高电平）应答，而通过产生一个停止条件应答时，顺序读取操作被终止。

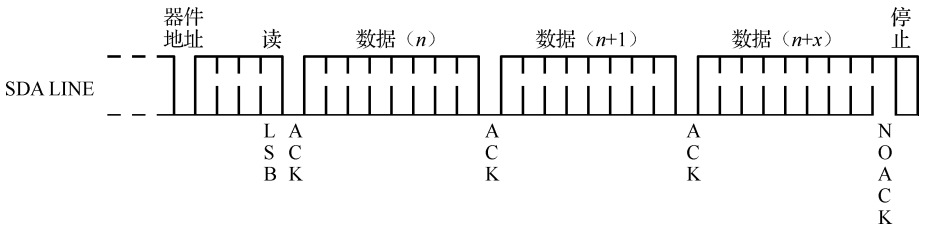



图 3.21　顺序读取时序

国信长天单片机竞赛实训平台的 AT24C02 电路图如图 3.22 所示。将器件地址输入端 A0～A2 接地；串行数据线 SCL、串行时钟 SDA 线分别和单片机引脚 P20、P21 相连；写保护 WP 进行接地，允许读写操作。



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

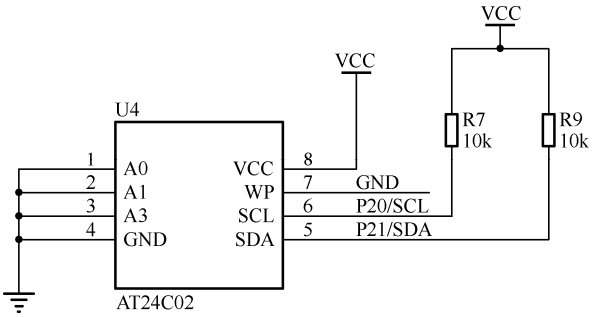



图 3.22　国信长天单片机竞赛实训平台 AT24C02 电路图

AT24C02 读写程序也在竞赛资源包“iic.h”和“iic.c”的基础上编写如下：

在“iic.h”中添加下列声明：

```
void EEPROM_Read(unsigned char* pucBuf, unsigned char addr,
  unsigned char num);
void EEPROM_Write(unsigned char* pucBuf, unsigned char addr,
  unsigned char num);
```

在“iic.c”中添加下列源代码：

```
#ifdef EEPROM_AT24C02
void EEPROM_Read(unsigned char* pucBuf, unsigned char addr,
  unsigned char num)
{
  IIC_Start();
  IIC_SendByte(0xa0);
  IIC_WaitAck();

  IIC_SendByte(addr);
  IIC_WaitAck();

  IIC_Start();
  IIC_SendByte(0xa1);
  IIC_WaitAck();

  while(num--)
  {
    *pucBuf++ = IIC_RecByte();
    if(num) IIC_SendAck(0);
    else IIC_SendAck(1);
  }
  IIC_Stop();
}

void EEPROM_Read(unsigned char* pucBuf, unsigned char addr,
  unsigned char num)
{
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
    IIC_Start();
    IIC_SendByte(0xa0);
    IIC_WaitAck();

    IIC_SendByte(addr);
    IIC_WaitAck();

    while(num--)
    {
      IIC_SendByte(*pucBuf++);
      IIC_WaitAck();
      IIC_Delay(200);
    }
    IIC_Stop();
    operate_delay(20);
  }
  #endif
```

AT24C02 读写程序的使用方法如下。

（1）在“iic.h”“iic.c”中添加 EEPROM_Read()和 EEPROM_Write()的声明及源代码。

（2）将“iic.c”添加到“Source Group 1”中。

（3）将“iic.h”包含在“main.h”中。

（4）在 main()中调用 EEPROM_Read()和 EEPROM_Write()。

（5）在“Options for Target”的“C51”标签中定义预处理符号“EEPROM_AT24C02”，参见图 3.16。

## 3.11 超声波传感器驱动程序设计

CX20106A 是一款红外线检波接收的专用芯片，常用于电视机红外遥控接收器。使用 CX20106A 作为超声波接收处理的典型电路（当 CX20106A 接收到 40kHz 的信号时，会在第 7 脚产生一个低电平下降脉冲，这个信号可以接到单片机的外部中断引脚作为中断信号输入）。使用 CX20106A 集成电路对接收探头收到的信号进行放大、滤波，其总放大增益 80dB。

考虑到红外遥控常用的载波频率 38kHz 与测距超声波频率 40kHz 较为接近，可以利用它作为超声波检测电路。实验证明，其具有很高的灵敏度和较强的抗干扰能力，但在使用时需注意：

（1）首先确定发射出的超声波的频率为标准的 40kHz（占空比 50%），并保证有足够的驱动电压。

（2）如果驱动的是开放式的非防水探头（就是铝外壳，探头表面有金属网，可以看到里面有一锥形的金属），则需使用反相器串联再并联做 BTL 推动（短距离测距，10Vpp 以上的电压即可）。

（3）如果使用的探头是全封闭的防水头，那就要注意必须有足够的驱动电压才能驱动得了探头（至少要 60Vpp，必须使用倒车雷达专用的中周变压器）。

（4）CX20106A 的第 5 脚的电阻决定接收的中心频率，200kΩ的电阻决定了接收的中心频率为 40kHz。

国信长天单片机竞赛实训平台的 CX20106A 红外检波接收电路图如图 3.23 所示，超声波发射电路图如图 3.24 所示，红外线和超声波选择电路图如图 3.25 所示。



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

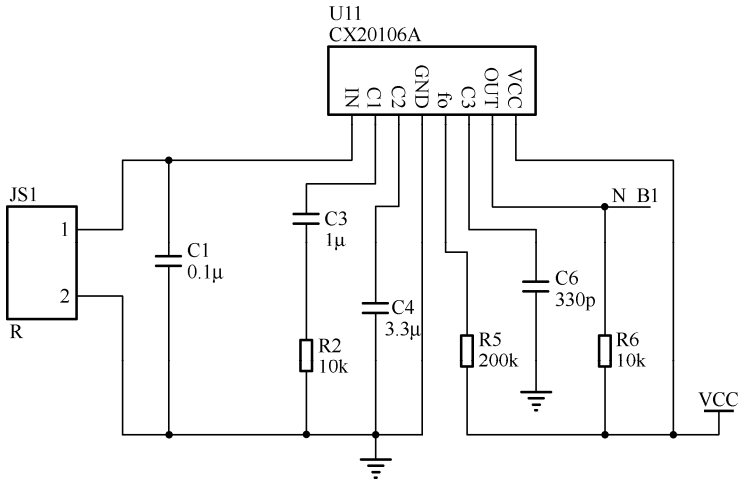



图 3.23　国信长天单片机竞赛实训平台 CX20106A 电路图

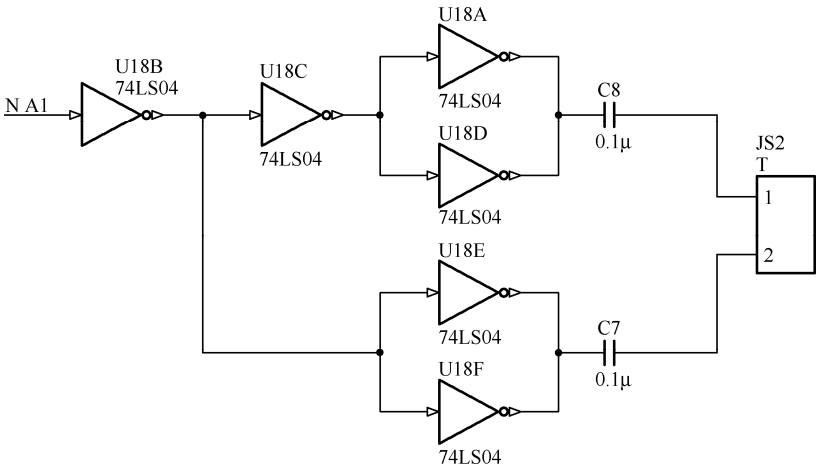



图 3.24　国信长天单片机竞赛实训平台超声波发射电路图

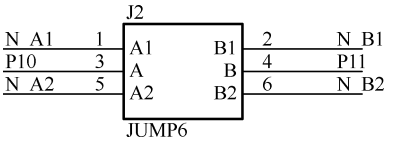



图 3.25　国信长天单片机竞赛实训平台红外线和超声波选择电路图

基于国信长天单片机竞赛实训平台的超声波测距程序设计如下：

超声波头文件“ultrasonic.h”内容如下：

```
#ifndef __Ultrasonic_H_
#define __Ultrasonic_H_
#include "STC15F2K60S2.H"

void Timer0Init(void);
unsigned char Wave_Recv(void);
#endif
```

超声波源文件“ultrasonic.c”内容如下：

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
#include "ultrasonic.h"

sbit TX = P1^0;                         // 发射引脚
sbit RX = P1^1;                         // 接收引脚

void Timer0Init(void)                   // 12us@12.000MHz
{
  AUXR &= 0x7F;                         // 定时器时钟 12T 模式
  TMOD &= 0xF0;                         // 设置定时器模式
  TL0 = 0xF4;                           // 设置定时初值
  TH0 = 0xFF;                           // 设置定时初值
  TF0 = 0;                              // 清除 TF0 标志
  TR0 = 0;                              // 定时器 0 停止
}

unsigned char Wave_Recv(void)
{
  unsigned char ucDist, ucNum = 10;

  TX = 0;
  TL0 = 0xF4;                           // 设置定时初值
  TH0 = 0xFF;                           // 设置定时初值
  TR0 = 1;                              // 定时器 0 计时
  // TX 引脚发送 40kHz 方波信号驱动超声波发送探头
  while(ucNum--)
  {
    while(!TF0);
    TX ^= 1;
    TF0 = 0;
  }
  TR0 = 0;
  TL0 = 0;                              // 设置定时初值
  TH0 = 0;                              // 设置定时初值
  TR0 = 1;
  while(RX && !TF0);                    // 等待收到脉冲
  TR0 = 0;
  if(TF0)                               // 发生溢出
  {
    TF0 = 0;
    ucDist = 255;
  }
  else                                 // 计算距离
    ucDist = ((TH0<<8)+TL0)*0.017;
  return ucDist;
}
```



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

**注意：**超声波测距时需要将 J2 的 1、3 和 2、4 短接。

## 3.12 频率测量程序设计

频率测量比较简单：将定时器 0 初始化为计数模式，对输入到 P34（T0）的脉冲进行计数，1s 的计数值即为频率值。

**注意：**频率测量时需要将 J3-15（SIGNAL）与 J3-16（P34）短接。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

# 第 4 章　单片机设计与开发项目往届试题解析

本章介绍单片机设计与开发项目往届试题解析，包括系统设计和系统实现等。为了排版和调试的需要，对试题的部分内容进行了修改。

## 4.1　第七届预赛试题——模拟风扇控制系统

模拟风扇控制系统能够模拟电风扇工作，通过按键控制风扇的转动速度和定时时间，数码管实时显示风扇的工作模式，动态倒计时显示剩余的定时时间，系统框图如图 4.1 所示。

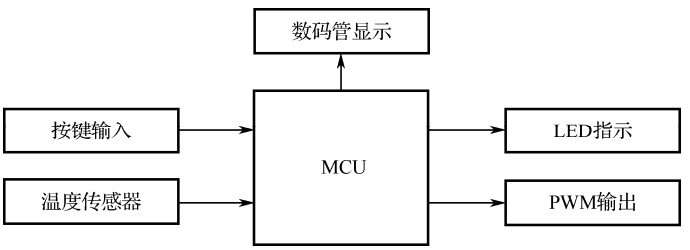



图 4.1　模拟风扇控制系统框图

模拟风扇控制系统功能描述如下。

**1）工作模式**

设备具有“睡眠风”“自然风”和“常风”三种工作模式，可以通过按键切换，通过单片机 P34 引脚输出脉宽调制信号控制电机运行，信号频率为 1kHz。

（1）“睡眠风”模式下，对应 PWM 占空比为 20%；

（2）“自然风”模式下，对应 PWM 占空比为 30%；

（3）“常风”模式下，对应 PWM 占空比为 70%。

**2）数码管显示**

数码管实时显示当前工作模式和剩余工作时间（倒计时），如图 4.2 所示。

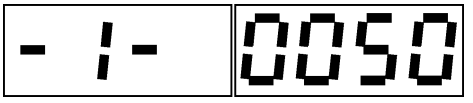



图 4.2　工作模式和剩余工作时间显示

其中前 3 位显示工作模式：1—睡眠风，2—自然风，3—常风；后 4 位显示剩余工作时间（单位：s）。

**3）按键控制**

（为了方便在线调试，将 S6 和 S7 更改为 S8 和 S9）。

（1）按键 S4 定义为工作模式切换按键，每次按下 S4，设备循环切换三种工作模式。

（2）按键 S5 定义为定时按键，每次按下 S5，定时时间在 0、1 和 2 分钟间切换，设备的剩余工作时间重置为当前定时时间，重新开始倒计时。设备剩余工作时间为 0 时，停止 PWM 信号输出。

（3）按键 S9 定义为“停止”按键，按下 S9 按键，立即清零剩余工作时间，PWM 信号停止输出，直到通过 S5 重新设置定时时间。

（4）按键 S8 定义为“室温”按键，按下 S8，通过数码管显示当前室温，数码管显示格式如



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

图 4.3 所示。再次按下 S8，返回工作模式和剩余工作时间显示。

图 4.3　当前室温显示

室温测量显示功能不应影响设备正在执行的 PWM 信号输出、停止、模式切换和计时等功能。

**4）LED 指示灯**

“睡眠风”模式下 L1 点亮，“自然风”模式下 L2 点亮，“常风”模式下 L3 点亮；按下停止按键或倒计时结束时，LED 全部熄灭。

### 4.1.1　系统设计

根据系统要求，可以得到系统详细框图如图 4.4 所示。单片机采集按键和温度数据，按要求将相关信息显示到数码管并点亮相关 LED 灯，定时器 0 产生 PWM 信号，并通过 P34 引脚输出控制风扇的转速。

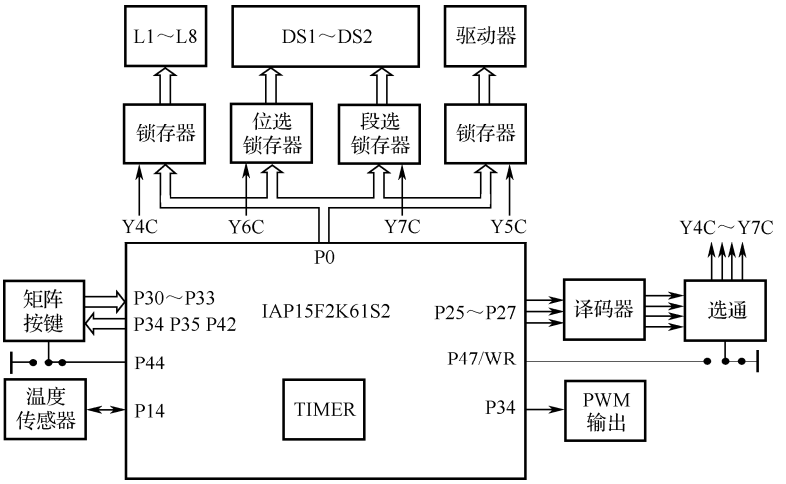



图 4.4　模拟风扇控制系统详细框图

为了产生频率为 1kHz（周期 1ms），占空比分别为 20%、30%和 70%的 PWM 信号，定时器 0 的周期设为 100μs，初始时 P34 输出高电平，定时到 200μs、300μs 或 700μs 时 P34 输出低电平。

模拟风扇控制系统主程序流程图如图 4.5 所示。主程序首先关闭外设，对定时器 1 和定时器 0 进行初始化，然后循环进行按键处理和数码管处理，其中数码管处理包含温度测量，定时器 1 中断处理实现秒计时，定时器 0 中断处理实现 PWM 输出和 LED 显示。

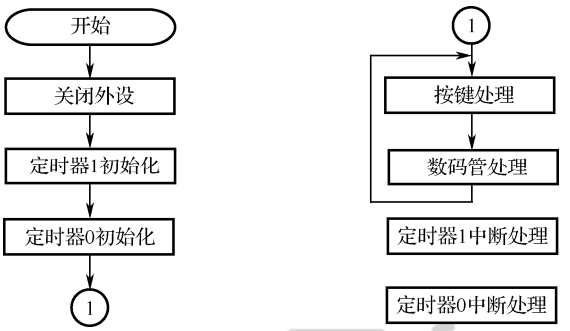



图 4.5　模拟风扇控制系统主程序流程图

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

模拟风扇控制系统“main.c 内容如下：

```
#include "tim.h"
#include "key.h"
#include "seg.h"
#include "stdio.h"
#include "onewire.h"

sbit Pwm_Out = P3^4;
unsigned int  uiSeg_Dly;
unsigned char ucTemp_Val, ucTemp_Flag;
unsigned char ucState, ucSec, ucLed=1;
unsigned char ucDelay, ucPwm, ucPlus=2;
unsigned char pucSeg_Buf[9], pucSeg_Code[8], ucSeg_Pos;
unsigned char ucKey_Dly, ucKey_Old;
unsigned long ulms;

void Key_Proc(void);
void Seg_Proc(void);

void Timer0Init(void)                   // 100 微秒@12.000MHz
{
  AUXR |= 0x80;                         // 定时器时钟 1T 模式
  TMOD &= 0xF0;                         // 设置定时器模式
  TL0 = 0x50;                           // 设置定时初值
  TH0 = 0xFB;                           // 设置定时初值
  TF0 = 0;                              // 清除 TF0 标志
  TR0 = 1;                              // 定时器 0 开始计时
  ET0 = 1;                              // 允许定时器 0 中断
}

void main(void)
{
  Cls_Peripheral();
  Timer1Init();
  Timer0Init();

  while(1)
  {
    Key_Proc();
    Seg_Proc();
  }
}

void Time_1(void) interrupt 3
{
```



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
  ulms++;
  if(++ucKey_Dly == 10)
    ucKey_Dly = 0;
  if(++uiSeg_Dly == 500)
    uiSeg_Dly = 0;
  if(!(ulms % 1000))
  {
    if(ucDelay) ucDelay--;
    else if(ucSec) ucSec--;
  }
  Seg_Disp(pucSeg_Code, ucSeg_Pos);
  if(++ucSeg_Pos == 8) ucSeg_Pos = 0;
}

void Key_Proc(void)
{
  unsigned char ucKey_Val, ucKey_Down;

  if(ucKey_Dly) return;
  ucKey_Dly = 1;

  ucKey_Val = Key_Read();
  ucKey_Down = ucKey_Val & (ucKey_Old ^ ucKey_Val);
  ucKey_Old = ucKey_Val;

  switch(ucKey_Down)
  {
    case 4:                                 // S4
      if(++ucState == 3)                    // 切换工作模式
        ucState = 0;
      switch(ucState)
      {
        case 0:
          ucLed |= 1; ucLed &= ~6;          // L1亮
          ucPlus = 2; break;
        case 1:
          ucLed |= 2; ucLed &= ~5;          // L2亮
          ucPlus = 3; break;
        case 2:
          ucLed |= 4; ucLed &= ~3;          // L3亮
          ucPlus = 7;
      }break;
    case 5:                                 // S5
      ucSec += 60;                          // 切换定时
      if(ucSec >= 180) ucSec = 0;
      else if(ucSec >= 120) ucSec = 120;
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
      else if(ucSec >= 60) ucSec = 60;
      ucDelay = 2;
      break;
    case 9:                          // S9
      ucSec = 0;                     // 清除定时
      break;
    case 8:                          // S8
      ucTemp_Flag ^= 1;              // 切换室温显示
  }
}

void Seg_Proc(void)
{
  if(uiSeg_Dly) return;
  uiSeg_Dly = 1;

  if(!ucTemp_Flag)
    sprintf(pucSeg_Buf, "-%1u- %04u", \
      (unsigned int)ucState+1, (unsigned int)ucSec);
  else
  {
    ucTemp_Val = rd_temperature()>>4;
    sprintf(pucSeg_Buf, "-4-  %2uC", (unsigned int)ucTemp_Val);
  }
  Seg_Tran(pucSeg_Buf, pucSeg_Code);
}

void Time_0(void) interrupt 1
{
  if(ucSec)
  {
    if(++ucPwm == 10) ucPwm = 0;
    if(!ucPwm)
    {
      Pwm_Out = 1;
      ucLed |= 0x80;
    }
    else if(ucPwm == ucPlus)
    {
      Pwm_Out = 0;
      ucLed &= 0x7F;
    }
  }
  else
    ucLed &= 0x7f;
  Led_Disp(ucLed);
}
```



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

### 4.1.2 系统实现

模拟风扇控制系统实现的主要步骤如下：

（1）在“D:\CT107D”文件夹中将“208_DS18B20”文件夹复制并重命名为“301_071”文件夹，进入“301_071”文件夹，双击工程文件“IAP15”打开工程。

（2）修改“main.c”的内容。

（3）编译并调试程序。

## 4.2 第七届决赛试题——电压频率采集设备

电压频率采集设备能够测量信号电压和频率，修改和存储工作参数，记录和查询事件等，系统框图如图 4.6 所示。

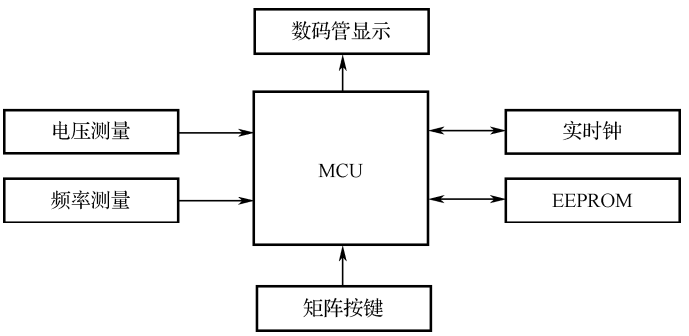



图 4.6 电压频率采集设备系统框图

电压频率采集设备功能描述如下：

**1）电压测量**

使用实训平台上 Rb2 电位器输出测试电压信号，信号输出范围为 0～5V。

**2）频率测量**

使用实训平台上 NE555 方波信号发生器产生用于频率测量的方波信号，信号频率范围为 500Hz ～ 20kHz，电位器 Rb3 用于调节信号频率，可使用跳线帽将单片机 P34 引脚与方波信号发生器输出引脚 NET_SIG 短接。

**3）实时钟功能**

读取 DS1302 实时钟芯片，并通过数码管显示时间，设备上电后初始时间配置为 23 时 59 分 55 秒。

**4）记录功能**

根据给定硬件的性能参数，设计合理的采样周期，当电压低于电压下限 $V_L$ 或高于电压上限 $V_H$ 时，将事件类型和事件时间保存到 EEPROM 中。设备中只需保存最近一次发生的电压波动情况，数据存储格式可自行定义。

**5）按键与显示控制**

按键功能定义如图 4.7 所示。

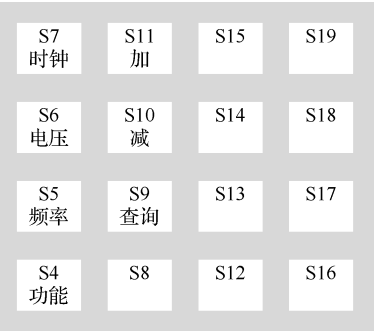



图 4.7 按键功能定义

（1）按键 S7 定义为时钟按键，按下后数码管显示当前时钟，显示界面如图 4.8 所示。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

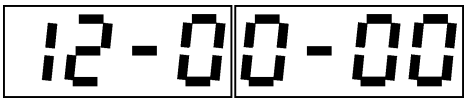



图 4.8　时钟显示界面

在时钟显示界面下，S4 为时钟调整按键，按下 S4，循环切换选择时、分、秒，对应的显示单元 1s 间隔亮灭，通过 S11 加按键和 S10 减按键调整当前选择的时间，再次按下按键 S7 返回时钟显示界面，完成时钟配置功能。

时钟调整状态下，加减按键可令当前选择调整的时间增加或减少 1 个单位。

（2）按键 S6 定义为电压测量按键，按下后启动电压测量功能，数码管显示信号电压（单位为 mV），格式如图 4.9 所示。

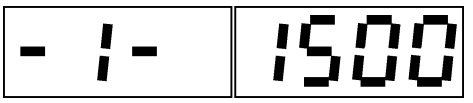



图 4.9　电压测量显示界面

在电压测量界面下，S4 为电压阈值调整按键，按下 S4，循环选择电压上限（前 4 位，默认值 2000mV）和电压下限（后 4 位，默认值 1000mV），对应的显示单元 1s 间隔亮灭，通过 S11-加按键和 S10-减按键调整电压阈值，再次按下按键 S6 返回到电压测量界面，完成电压阈值配置功能并将新的参数写入 EERPOM。

电压阈值调整状态下，加减按键可令当前选择调整的阈值增加或减少 500mV。

（3）按键 S5 定义为频率测量按键，按下后启动频率测量功能，设备采集输入到 P34 引脚的信号频率，数码管切换到频率测量显示界面。在频率测量功能下，按键 S4 定义为周期/频率按键，可以进行频率（单位 Hz）和周期（单位 μs）的显示切换，显示界面如图 4.10 所示（1000Hz 或 1000μs）。

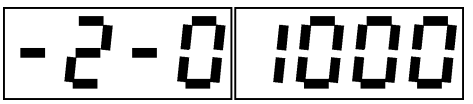



图 4.10　频率测量显示界面

（4）S9 定义为查询按键，按下后显示最近一次电压波动发生的时间和波动类型，显示格式如图 4.11 和图 4.12 所示。

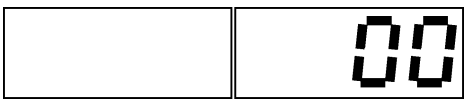



图 4.11　事件查询界面（事件类型）

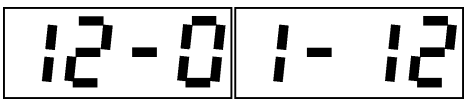



图 4.12　事件查询界面（发生时间）

在事件查询界面下，S4 可切换显示事件类型和事件时间，“00”表示低于下限事件，“01”表示超出上限事件。

**6）说明**

（1）最近一次电压波动发生时间和类型、电压上限 $V_H$、电压下限 $V_L$ 保存在 EEPROM 中，设备重新上电后，能够从 EEPROM 中获取参数和数据。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

（2）建立一个以准考证号命名的 txt 文档，写出设定的单片机内部振荡器频率，保存在考生文件夹中。

### 4.2.1 系统设计

电压频率采集设备的详细框图如图 4.13 所示。电压信号通过 ADC 转换后输出到单片机，555 方波输出至单片机 P34 引脚，由计数器 0 计数得到方波频率。24C02 作为 EEPROM 芯片存储记录信息，DS1302 实时钟芯片产生实时钟信息，单片机从 DS1302 采集时钟信息，从键盘采集功能要求，并将时钟等信息按要求显示到数码管。

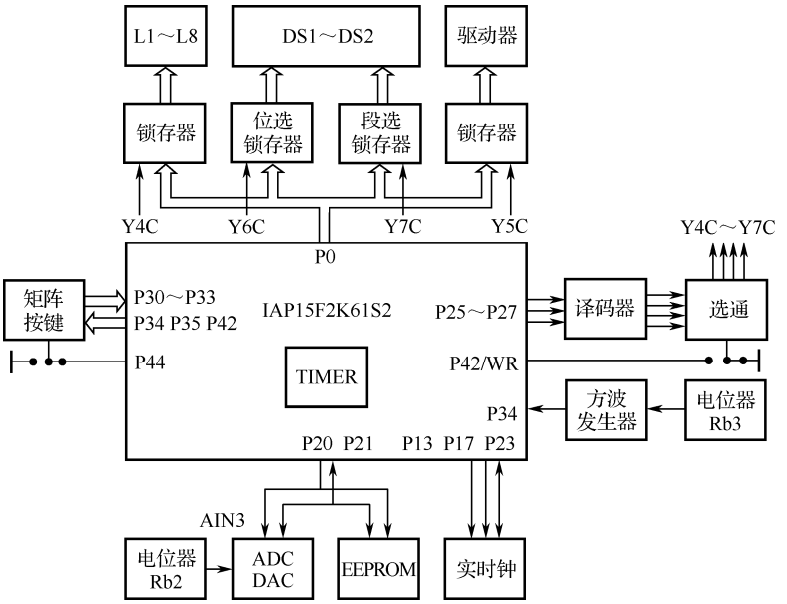



图 4.13　电压频率采集设备详细框图

电压频率采集设备主程序流程图如图 4.14 所示。主程序首先关闭外设、初始化定时器 1 和定时器 0，然后设置时钟并读取阈值，最后进行按键处理和数码管处理，其中按键处理中包含读取事件和保存阈值，数码管处理中包含读取时钟、电压测量和保存事件等，频率测量在定时器 1 中断处理中实现。

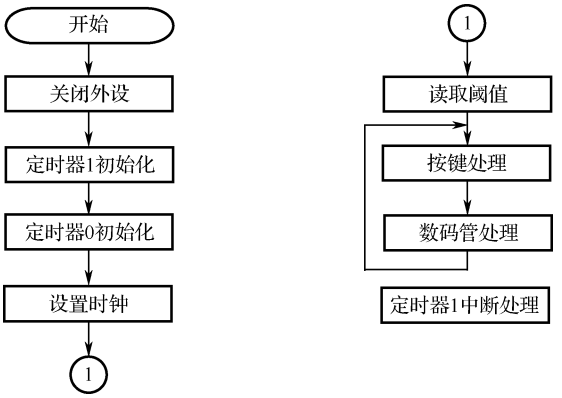



图 4.14　电压频率采集设备主程序流程图

电压频率采集设备“main.c”内容如下：

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
#include "tim.h"
#include "key.h"
#include "seg.h"
#include "stdio.h"
#include "ds1302.h"
#include "iic.h"

unsigned int  uiSeg_Dly;
unsigned int  uiVol_Val, uiFre_Val;
unsigned int  uiVol_Low, uiVol_High;
unsigned char pucRtc[3] = {23, 59, 50};
unsigned char ucState, ucSec, pucEvent[4];
unsigned char pucSeg_Buf[9], pucSeg_Code[8], ucSeg_Pos;
unsigned char ucKey_Dly, ucKey_Old;
unsigned long ulms;

void Timer0Init(void)                   // 16 位计数器模式
{
  AUXR &= 0x7F;                         // 定时器时钟 12T 模式
  TMOD |= 5;                            // 设置 16 位计数器模式
  TL0 = 0;                              // 设置计数初值
  TH0 = 0;                              // 设置计数初值
  TF0 = 0;                              // 清除 TF0 标志
  TR0 = 1;                              // 开始计数
}

void Key_Proc(void);
void Seg_Proc(void);

void main(void)
{
  Cls_Peripheral();
  Timer1Init();
  Timer0Init();

  Set_RTC(pucRtc);
  EEPROM_Read(pucSeg_Buf, 0, 4);
  uiVol_Low = (pucSeg_Buf[1]<<8) + pucSeg_Buf[0];
  uiVol_High = (pucSeg_Buf[3]<<8) + pucSeg_Buf[2];
  if(uiVol_Low > 5000) uiVol_Low = 1000;
  if(uiVol_High > 5000) uiVol_High = 2000;

  while(1)
  {
```



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
    Key_Proc();
    Seg_Proc();
  }
}

void Time_1(void) interrupt 3
{
  ulms++;
  if(++ucKey_Dly == 10)
    ucKey_Dly = 0;
  if(++uiSeg_Dly == 500)
    uiSeg_Dly = 0;
  if(!(ulms % 1000))
  {
    ucSec++;
    uiFre_Val = (TH0<<8)+TL0;
    TH0 = 0;
    TL0 = 0;
  }
  Seg_Disp(pucSeg_Code, ucSeg_Pos);
  if(++ucSeg_Pos == 8) ucSeg_Pos = 0;
}
```

为了方便在线调试，将按键功能重新定义如图 4.15 所示。

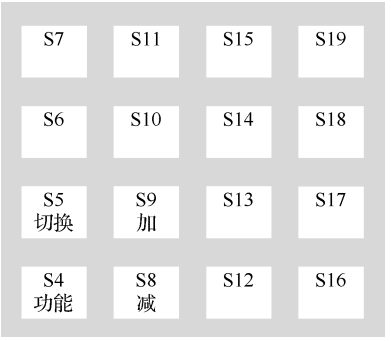



图 4.15　按键功能重新定义

其中，S5 用于切换显示时钟、显示电压、显示频率或显示事件等功能。

S4 用于 4 大功能下的功能选择：

显示时钟时，S4 用于选择显示时钟、设置时、设置分或设置秒。

显示电压时，S4 用于选择显示电压、设置电压上限、设置电压下限。

显示频率时，S4 用于选择显示频率或周期。

显示事件时，S4 用于选择显示事件类型或时间。

S9 和 S8 分别实现设置时钟和电压上下限时的加减功能。

电压频率采集设备的状态图如图 4.16 所示。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

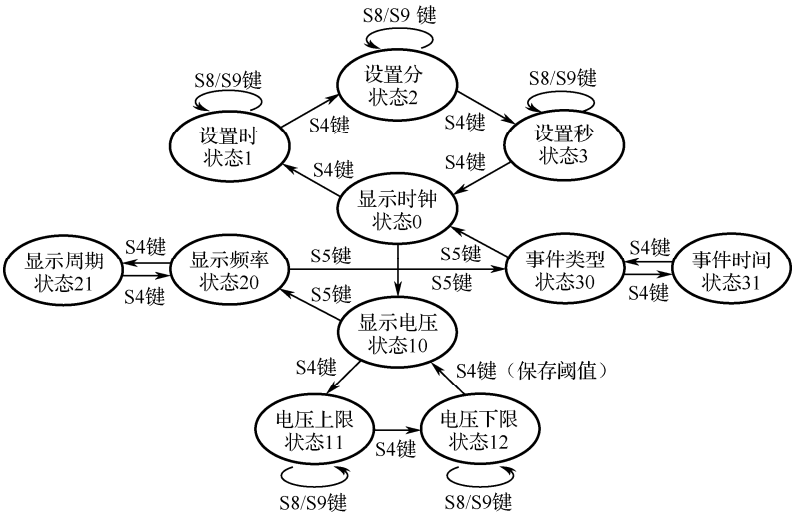



图 4.16　电压频率采集设备状态图

基于电压频率采集设备状态图，设计按键处理和数码管处理程序如下：

```
void Key_Proc(void)
{
  unsigned char ucKey_Val, ucKey_Down;

  if(ucKey_Dly) return;
  ucKey_Dly = 1;

  ucKey_Val = Key_Read();
  ucKey_Down = ucKey_Val & (ucKey_Old ^ ucKey_Val);
  ucKey_Old = ucKey_Val;

  switch(ucKey_Down)
  {
    case 5:                              // S5：切换显示时钟、电压、频率或事件
      ucState = (ucState&0xf0)+0x10;
      if(ucState >= 0x40) ucState = 0;
      if(ucState == 0x30)
      {                                  // 读取事件
        EEPROM_Read(pucEvent, 4, 4);
        if(pucEvent[0] > 2) pucEvent[0] = 0;
        if(pucEvent[1] > 24) pucEvent[1] = 0;
        if(pucEvent[2] > 60) pucEvent[2] = 0;
        if(pucEvent[3] > 60) pucEvent[3] = 0;
      }break;
    case 4:                              // S4：功能选择
      switch(ucState&0xf0)
      {
        case 0:                          // 选择显示时钟、设置时、设置分或设置秒
          if(++ucState == 4)
```



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
      {
        ucState = 0;
        Set_RTC(pucRtc);            // 设置时钟
      }break;
    case 0x10:                      // 选择显示电压、设置电压上限或下限
      if(++ucState == 0x13)
      {
        ucState = 0x10;
        // 保存阈值
        pucSeg_Buf[0] = uiVol_Low & 0xff;
        pucSeg_Buf[1] = uiVol_Low >> 8;
        pucSeg_Buf[2] = uiVol_High & 0xff;
        pucSeg_Buf[3] = uiVol_High >> 8;
        EEPROM_Write(pucSeg_Buf, 0, 4);
      }break;
    case 0x20:                      // 选择显示频率或周期
      if(++ucState == 0x22) ucState = 0x20;
      break;
    case 0x30:                      // 选择显示事件类型或时间
      if(++ucState == 0x32) ucState = 0x30;
  }break;
case 9:                             // S9：参数加
  switch(ucState)
  {
    case 1:
      if(++pucRtc[0] >= 24) pucRtc[0] = 0;
      break;
    case 2:
      if(++pucRtc[1] >= 60) pucRtc[1] = 0;
      break;
    case 3:
      if(++pucRtc[2] >= 60) pucRtc[2] = 0;
      break;
    case 0x11:
      if(uiVol_High <= 4500) uiVol_High += 500;
      break;
    case 0x12:
      if(uiVol_High - uiVol_Low > 500) uiVol_Low += 500;
  }break;
case 8:                             // S8：参数减
  switch(ucState)
  {
    case 1:
      if(pucRtc[0]) pucRtc[0]--;
      else pucRtc[0] = 23;
      break;
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
        case 2:
          if(pucRtc[1]) pucRtc[1]--;
          else pucRtc[1] = 59;
          break;
        case 3:
          if(pucRtc[2]) pucRtc[2]--;
          else pucRtc[2] = 59;
          break;
        case 0x11:
          if(uiVol_High - uiVol_Low > 500) uiVol_High -= 500;
          break;
        case 0x12:
          if(uiVol_Low >= 500) uiVol_Low -= 500;
      }
    }
    Led_Disp(ucState+0x11);
}

void Seg_Proc(void)
{
    if(uiSeg_Dly) return;
    uiSeg_Dly = 1;

    switch(ucState&0xf0)
    {
      case 0:                                  // 显示时钟
        if(!ucState) Read_RTC(pucRtc);
        sprintf(pucSeg_Buf, "%02u-%02u-%02u", (unsigned int)pucRtc[0],\
          (unsigned int)pucRtc[1], (unsigned int)pucRtc[2]);
        if(ucSec&1)
          switch(ucState)
          {
            case 1:                            // 熄灭时
              pucSeg_Buf[0] = 0xff;
              pucSeg_Buf[1] = 0xff;
              break;
            case 2:                            // 熄灭分
              pucSeg_Buf[3] = 0xff;
              pucSeg_Buf[4] = 0xff;
              break;
            case 3:                            // 熄灭秒
              pucSeg_Buf[6] = 0xff;
              pucSeg_Buf[7] = 0xff;
           }break;
      case 0x10:
        uiVol_Val = PCF8591_Adc()*19.6;        // 5000/255=19.6
```



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
  if((uiVol_Val<uiVol_Low)||(uiVol_Val>uiVol_High))
  {                                    // 保存事件
    if(uiVol_Val<uiVol_Low) pucSeg_Buf[0] = 0;
    else pucSeg_Buf[0] = 1;
    pucSeg_Buf[1] = pucRtc[0];
    pucSeg_Buf[2] = pucRtc[1];
    pucSeg_Buf[3] = pucRtc[2];
    EEPROM_Write(pucSeg_Buf, 4, 4);
  }
  if(ucState == 0x10)
    sprintf(pucSeg_Buf, "-1- %04u", uiVol_Val);
  else
  {                                    // 显示阈值
    sprintf(pucSeg_Buf, "%04u%04u", uiVol_High, uiVol_Low);
    if(ucSec&1)
      if(ucState == 0x11)
      {                                // 熄灭上限
        pucSeg_Buf[0] = 0xff;
        pucSeg_Buf[1] = 0xff;
        pucSeg_Buf[2] = 0xff;
        pucSeg_Buf[3] = 0xff;
      }
      else
      {                                // 熄灭下限
        pucSeg_Buf[4] = 0xff;
        pucSeg_Buf[5] = 0xff;
        pucSeg_Buf[6] = 0xff;
        pucSeg_Buf[7] = 0xff;
      }
  }break;
case 0x20:
  switch(ucState)
  {
    case 0x20:                         // 显示频率
      sprintf(pucSeg_Buf, "-2-%05u", (unsigned int)uiFre_Val);
      break;
    case 0x21:                         // 显示周期
      sprintf(pucSeg_Buf, "-3-%05.0f", (float)1000000.0/uiFre_Val);
  }break;
case 0x30:
  switch(ucState)
  {
    case 0x30:                         // 显示事件类型
      sprintf(pucSeg_Buf, "     %02u", (unsigned int)pucEvent[0]);
      break;
    case 0x31:                         // 显示事件时间
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
            sprintf(pucSeg_Buf, "%02u-%02u-%02u", (unsigned int)pucEvent[1],\
              (unsigned int)pucEvent[2], (unsigned int)pucEvent[3]);
        }
      }
      Seg_Tran(pucSeg_Buf, pucSeg_Code);
    }
```

### 4.2.2 系统实现

电压频率采集设备实现的主要步骤如下:

（1）在“D:\CT107D”文件夹中将“207_DS1302”文件夹复制并重命名为“302_072”文件夹，进入“302_072”文件夹，双击工程文件“IAP15”打开工程。

（2）将“iic.c”添加到“Source Group 1”中。

（3）修改“main.c”的内容。

（4）在“Options for Target”的“C51”标签中定义预处理符号“PCF8591_ADC EEPROM_AT24C02”。

（5）编译并调试程序。

**注意：**频率测量时需要将 J3-15（SIGNAL）与 J3-16（P34）短接。

## 4.3 第八届预赛试题——电子钟

电子钟系统框图如图 4.17 所示。

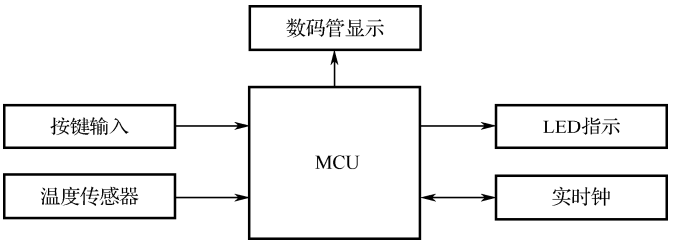



图 4.17　电子钟系统框图

电子钟系统功能描述如下。

**1）初始化**

（1）关闭蜂鸣器、继电器等无关外设。

（2）设备初始化时钟为 23 时 59 分 50 秒，闹钟提醒时间 0 时 0 分 0 秒。

**2）显示功能**

（1）时间显示格式如图 4.18 所示。

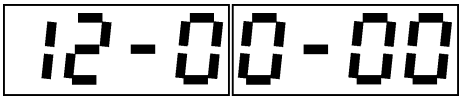



图 4.18　时间显示格式

（2）温度显示格式如图 4.19 所示。

图 4.19　温度显示格式

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

**3）按键功能（为了方便在线调试，对按键功能进行了调整）**

（1）按键 S5 定义为时钟设置按键，通过该按键可切换选择待调整的时、分、秒，当前选择的显示单元以 1s 为间隔亮灭，时、分、秒的调整需注意数据边界属性。设置秒后再按 S5 返回时间显示。

（2）按键 S4 定义为闹钟设置按键，通过该按键可进入闹钟时间设置功能，数码管显示当前设定的闹钟时间。设置秒后再按 S4 返回时间显示。

（3）按键 S9 定义为加按键，在时钟设置或闹钟设置状态下， 每次按下该按键当前选择的单元（时、分或秒）增加 1 个单位。

（4）按键 S8 定义为减按键，在时钟设置或闹钟设置状态下， 每次按下该按键当前选择的单元（时、分或秒）减少 1 个单位。

（5）按键功能说明：

按键 S9 和 S8 的加减功能只在时钟设置或闹钟设置状态下有效。在时钟显示状态下，按下 S8 按键，显示温度数据，松开按键，返回时间显示。

**4）闹钟提示功能**

（1）指示灯 L1 以 0.2s 为间隔闪烁，持续 5s。

（2）在闹钟提示状态下，按下任意按键，关闭闪烁提示功能。

### 4.3.1 系统设计

电子钟系统详细框图如图 4.20 所示。单片机从 DS1302 采集实时钟信息，从 DS18B02 采集温度，从键盘采集功能要求，将时钟和温度信息按要求显示到数码管，闹钟发生时刻 L1 闪烁。

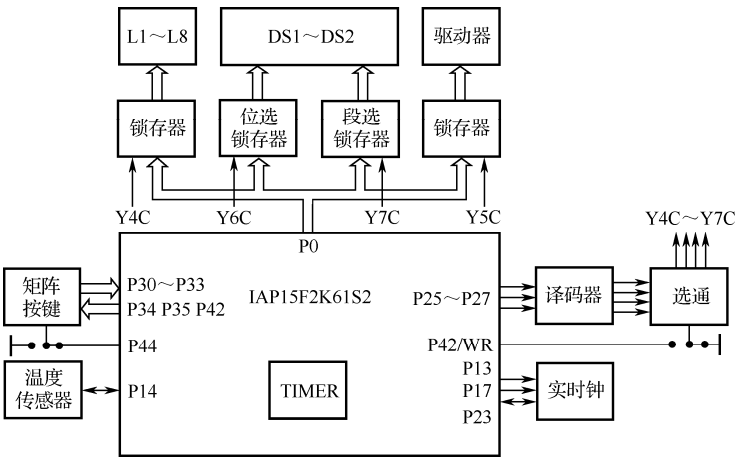



图 4.20　电子钟系统详细框图

电子钟系统主程序流程图如图 4.21 所示。主程序首先关闭外设、对定时器 1 进行初始化并设置时钟，然后进行按键处理和数码管处理，其中数码管处理中包含读取时钟和温度测量，LD1 闪烁在定时器 1 中断处理中实现。

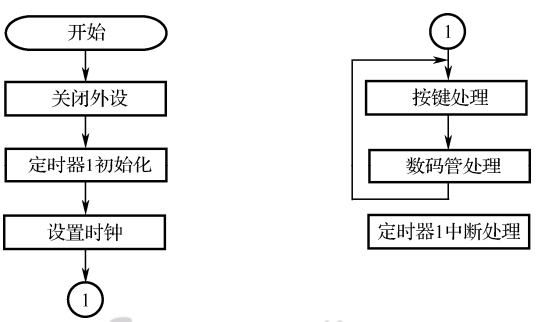



图 4.21　电子钟系统主程序流程图

电子钟系统“main.c”内容如下：

```
#include "tim.h"
#include "key.h"
#include "seg.h"
```

电子工业出版社 PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
#include "stdio.h"
#include "ds1302.h"
#include "onewire.h"

unsigned int  uiSeg_Dly;
unsigned char pucRtc[3] = {23,59,50};
unsigned char ucTemp_Val, ucTemp_Flag;
unsigned char pucAlarm[3] = {0,0,0}, ucAlarm_Num;
unsigned char ucState, ucSec, ucLed;
unsigned char pucSeg_Buf[9], pucSeg_Code[8], ucSeg_Pos;
unsigned char ucKey_Dly, ucKey_Old;
unsigned long ulms;

void Key_Proc(void);
void Seg_Proc(void);

void main(void)
{
  Cls_Peripheral();
  Timer1Init();
  Set_RTC(pucRtc);

  while(1)
  {
    Key_Proc();
    Seg_Proc();
  }
}

void Time_1(void) interrupt 3
{
  ulms++;
  if(++ucKey_Dly == 10) ucKey_Dly = 0;
  if(++uiSeg_Dly == 500) uiSeg_Dly = 0;
  if(ucAlarm_Num)
  {
    if(!(ulms % 200))
    {
      if(--ucAlarm_Num) ucLed ^= 1;
      else ucLed = 0;
      Led_Disp(ucLed);
    }
  }
  if(!(ulms % 1000))
    ucSec++;
  Seg_Disp(pucSeg_Code, ucSeg_Pos);
```



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
  if(++ucSeg_Pos == 8) ucSeg_Pos = 0;
}

void Key_Proc(void)
{
  unsigned char ucKey_Val, ucKey_Down;

  if(ucKey_Dly) return;
  ucKey_Dly = 1;

  ucKey_Val = Key_Read();
  ucKey_Down = ucKey_Val & (ucKey_Old ^ ucKey_Val);
  ucKey_Old = ucKey_Val;

  switch(ucKey_Down)
  {
    case 5:                          // S5
      if(!ucState) ucState += 0x10;
      else if(ucState < 0x20)
        if(++ucState == 0x13)
        {
          ucState = 0;
          Set_RTC(pucRtc);
        }break;
    case 4:                          // S4
      if(!ucState) ucState += 0x20;
      else if(ucState >= 0x20)
        if(++ucState == 0x23) ucState = 0;
      break;
    case 9:                          // S9
      if(!ucState) break;
      switch(ucState)
      {
        case 0x10:
          if(++pucRtc[0] == 24) pucRtc[0] = 0;
          break;
        case 0x11:
          if(++pucRtc[1] == 60) pucRtc[1] = 0;
          break;
        case 0x12:
          if(++pucRtc[2] == 60) pucRtc[2] = 0;
          break;
        case 0x20:
          if(++pucAlarm[0] == 24) pucAlarm[0] = 0;
          break;
        case 0x21:
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
          if(++pucAlarm[1] == 60) pucAlarm[1] = 0;
          break;
        case 0x22:
          if(++pucAlarm[2] == 60) pucAlarm[2] = 0;
      }break;
    case 8:                          // S8
      if(!ucState) break;
      switch(ucState)
      {
        case 0x10:
          if(!pucRtc[0]) pucRtc[0] = 23;
          else pucRtc[0]--;
          break;
        case 0x11:
          if(!pucRtc[1]) pucRtc[1] = 59;
          else pucRtc[1]--;
          break;
        case 0x12:
          if(!pucRtc[2]) pucRtc[2] = 59;
          else pucRtc[2]--;
          break;
        case 0x20:
          if(!pucAlarm[0]) pucAlarm[0] = 23;
          else pucAlarm[0]--;
          break;
        case 0x21:
          if(!pucAlarm[1]) pucAlarm[1] = 59;
          else pucAlarm[1]--;
          break;
        case 0x22:
          if(!pucAlarm[2]) pucAlarm[2] = 59;
          else pucAlarm[2]--;
      }
  }
  if(!ucState && (ucKey_Old == 8))
    ucTemp_Flag = 1;
  if(!ucState && !ucKey_Old)
    ucTemp_Flag = 0;

  if(ucKey_Down && ucAlarm_Num)
  {
    ucAlarm_Num = 0;
    Led_Disp(0);
  }
}
```



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
void Seg_Proc(void)
{
  if(uiSeg_Dly) return;
  uiSeg_Dly = 1;

  if(!ucTemp_Flag)
  {
    if(!ucState)
    {
      Read_RTC(pucRtc);
      if((pucRtc[0]==pucAlarm[0]) && (pucRtc[1]==pucAlarm[1])\
        && (pucRtc[2]==pucAlarm[2]))
        ucAlarm_Num = 25;
    }
    if(ucState < 0x20)
      sprintf(pucSeg_Buf, "%02d-%02d-%02d", (unsigned int)pucRtc[0],\
        (unsigned int)pucRtc[1], (unsigned int)pucRtc[2]);
    else
      sprintf(pucSeg_Buf, "%02d-%02d-%02d", (unsigned int)pucAlarm[0],\
        (unsigned int)pucAlarm[1], (unsigned int)pucAlarm[2]);
    if(ucSec&1)
      switch(ucState)
      {
        case 0x10:
        case 0x20:
          pucSeg_Buf[0] = 0xff;
          pucSeg_Buf[1] = 0xff;
          break;
        case 0x11:
        case 0x21:
          pucSeg_Buf[3] = 0xff;
          pucSeg_Buf[4] = 0xff;
          break;
        case 0x12:
        case 0x22:
          pucSeg_Buf[6] = 0xff;
          pucSeg_Buf[7] = 0xff;
      }
  }
  else
  {
    ucTemp_Val = rd_temperature()>>4;
    sprintf(pucSeg_Buf, "    %02dC", (unsigned int)ucTemp_Val);
  }
  Seg_Tran(pucSeg_Buf, pucSeg_Code);
}
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

### 4.3.2 系统实现

电子钟系统实现的主要步骤如下：

（1）在“D:\CT107D”文件夹中将“207_DS1302”或“208_DS18B20”文件夹复制并重命名为“303_081”文件夹，进入“303_081”文件夹，双击工程文件“IAP15”打开工程。

（2）将“onewire.c”或“ds1302.c”添加到“Source Group 1”中。

（3）修改“main.c”的内容。

（4）编译并调试程序。

### 4.3.3 客观题

（1）IAP15F2K61S2 单片机的定时器 0 具有_____种工作模式，当采用外部 12MHz 晶振时，定时器最大定时长度为_____μs。

（2）运放电路如图 4.22 所示，其输入电压 $U_{I1}$ 和 $U_{I2}$ 分别为 0.1V 和 0.2V，输出电压 $U_O$ 的值为_____V。

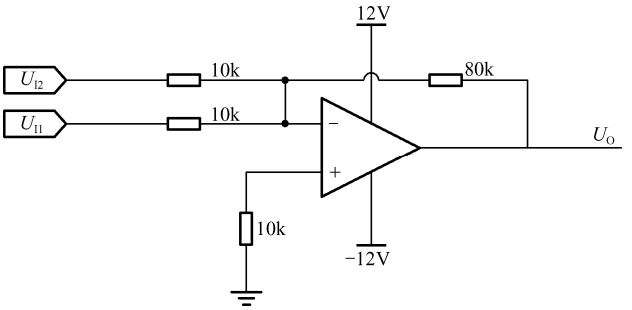



图 4.22　运放电路

（3）当电路中的有用信号为某一固定频率时，宜选用_____滤波器，直流电源的滤波电路宜选用_____滤波器。

A．带阻　　B．低通　　C．高通　　D．带通

（4）能够实现线与功能的门电路是_____。

A．与非门　　B．或非门　　C．异或门　　D．OC 门

（5）逻辑表达式 F=A$\overline{B}$+BC+ABCD+$\overline{B}$ 的最简式为_____。

A．C+D　　B．C　　C．$\overline{B}$+C　　D．$\overline{A}$+C

（6）MCS-51 单片机在同一优先级的中断源同时申请中断时，单片机首先响应下列_____中断源的请求。

A．串口中断　　B．定时器 0 中断

C．定时器 1 中断　　D．外部中断 0

（7）8051 单片机的 P0 口，当使用外部存储器时它是一个_____。

A．传输高 8 位地址口　　B．传输低 8 位地址口

C．传输高 8 位数据口　　D．传输低 8 位地址/数据口

（8）数码管动态扫描的程序设计一般需要“消隐”动作，才能保证显示效果清晰，下面基于实训平台的数码管显示代码片段中第_____行是用来实现“消隐”功能的。

```
1 void display(void)
2 {
3    XBYTE[0xE000] = 0xff;
```



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
4    XBYTE[0xC000] = (1<<dspcom);
5    XBYTE[0xE000] = tab[dspbuf[dspcom]];
6
7    if(++dspcom == 8)
8       dspcom = 0;
9 }
```

A．3　　B．4　　C．5　　D．8

（9）使用 Keil μVision 编写 51 单片机的 C 程序时，若定义一个变量 x，并由编译器将其分配到外部 RAM 中，应定义_____语句。

A．code unsigned char x;　　B．pdata unsigned char x;

C．idata unsigned char x;　　D．xdata unsigned char x;

（10）关于单片机下列_____是错误的。

A．IAP15F2K61S2 单片机复位后，P0～P3 口状态为低电平

B．具有 PWM 功能的单片机可通过滤波器实现 DAC 功能

C．IAP15F2K61S2 可以使用内部 RC 振荡器，也可以使用外部晶振工作

D．所有单片机的程序下载都需要冷启动过程

参考答案与评分标准：

（1）4，65536（4 分，每空 2 分）　（2）-2.4（2 分）

（3）D，B（3 分，每项 1.5 分）　（4）D（3 分）

（5）C（3 分）　（6）D（3 分）　（7）D（3 分）

（8）A（3 分）　（9）D（3 分）　（10）AD（3 分）

## 4.4　第八届决赛试题——超声波测距机

超声波测距机硬件框图如图 4.23 所示。

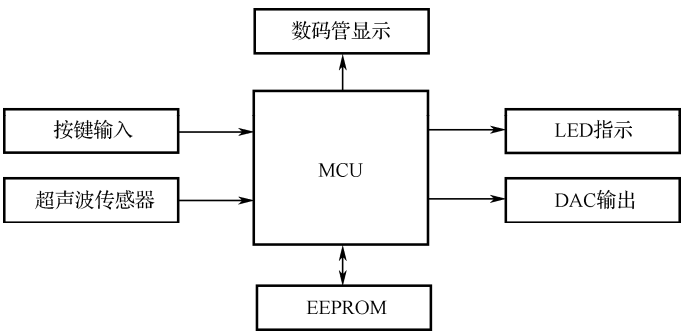



图 4.23　超声波测距机硬件框图

超声波测距机功能描述如下。

**1）初始化**

（1）关闭与试题要求实现功能无关的外围设备。

（2）设备上电初始状态下，处于测距显示界面。

**2）显示功能**

（1）测距显示界面如图 4.24 所示。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

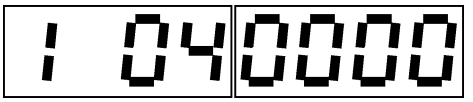


图 4.24　测距显示界面

- 第 1 位表示操作：0—无操作（计算结果显示上一次的测量数据），1—加操作（计算结果显示上一次测量与本次测量的和），设备上电后默认为 0。
- 第 3～5 位显示上次测量结果或上次与本次测量的和（单位：cm）。
- 第 6～8 位显示本次测量结果（单位：cm）。

（2）数据回显界面如图 4.25 所示。

图 4.25　数据回显界面

- 前 2 位显示数据编号。设备保存最近 10 次的测量结果，测量超过 10 次，则循环覆盖。数据编号用于说明当前数码管显示的数据是第几次测量的结果。数据编号与数码管显示的测量结果应具有对应关系。
- 后 3 位显示测量结果（单位：cm）。

（3）参数设置界面如图 4.26 所示。

图 4.26　参数设置界面

- 第 1 位为界面标识（字符 F）。
- 后 2 位显示测量盲区（单位：cm），用于校准 DAC 输出电压值，可通过按键进行参数设置。

**3）按键功能（为了方便在线调试，对按键功能进行了调整）**

（1）S4 按键——启动测量：按下按键启动一次测距过程，通过数码管显示本次的测量结果。

（2）S5 按键——数据回显：按下按键进入数据回显界面，再次按下切换回测距显示界面。

（3）S9 按键——参数设置：按下按键进入参数设置界面，再次按下保存当前设置并切换回测距显示界面。

（4）S8 按键：在测距显示界面下，S8 定义为操作功能，切换选择加操作或无操作。在数据回显界面下，S8 定义为翻页功能，查看 10 次以内的历史数据。在参数设置界面下，S8 定义为参数调整按键，每次按下 S8，测量盲区步进 10cm，参数可在 0～90 之间循环切换调整。

**4）模拟信号输出功能**

通过 PCF8591 实现 DAC 输出功能，DAC 输出的电压值取决于超声波测距的结果，关系如下：

当 $S \leqslant S_0$ 时，$V_{OUT}$=0V；

当 $S > S_0$ 时 $V_{OUT}=(S-S0)\times 0.02$V。

其中 $S_0$ 为测量盲区，$S$ 为当前测量结果，$V_{OUT}$ 为 DAC 输出的电压，当 $V_{OUT}$>5V 时，可当作异常状态处理，$V_{OUT}$ 输出 5V。

PCF8591 DAC 输出引脚为实训平台 J3-19 脚，标号为 D/A。

**5）数据存储功能**

测量盲区和最近 10 次的直接测量结果应保存在 EEPROM 中。



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

6）指示灯功能

（1）L1——测量完成指示灯：每次测量完成后以 0.2s 亮灭 10 次。

（2）L7——数据回显指示灯：进入数据回显界面时，指示灯点亮，退出后指示灯熄灭。

（3）L8——参数设置指示灯：进入参数设置界面时，指示灯点亮，退出后指示灯熄灭。

### 4.4.1 系统设计

超声波测距机系统框图如图 4.27 所示。超声波测距前，将 J2 的 1 脚和 3 脚短接使 P10 成为发射引脚、2 脚和 4 脚短接使 P11 成为接收引脚；测距时，单片机发送测速脉冲，接收返回脉冲，从 2 个脉冲时差中计算得出距离值。单片机从键盘采集功能要求，按要求测距并控制 LED 灯的点亮与熄灭。24C02 作为 EEPROM 存储记录信息，将相关数据显示在数码管上，测得距离值按要求用 DAC 转化为模拟信号输出在 J3-19 脚。

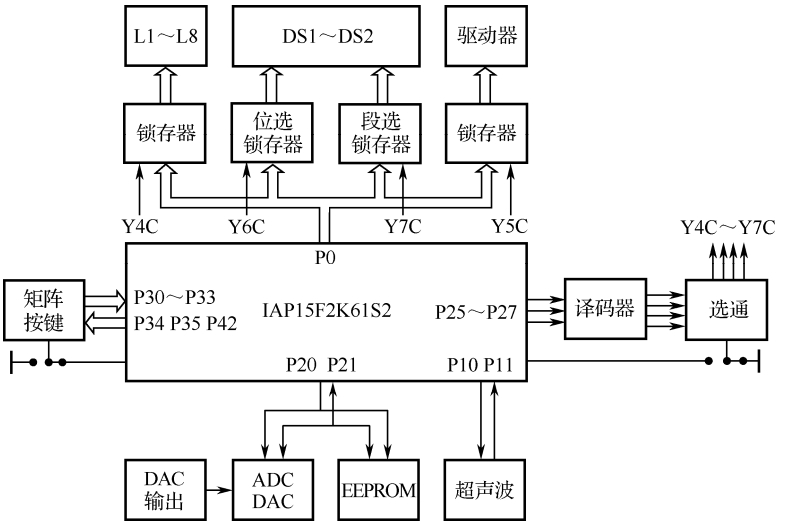



图 4.27　超声波测距机系统框图

系统主程序流程图如图 4.28 所示。主程序首先关闭外设，对定时器 1 和定时器 0 进行初始化，然后读取测量盲区和结果，最后进行按键处理和数码管处理，其中按键处理包括超声波测量和 DAC 输出等。

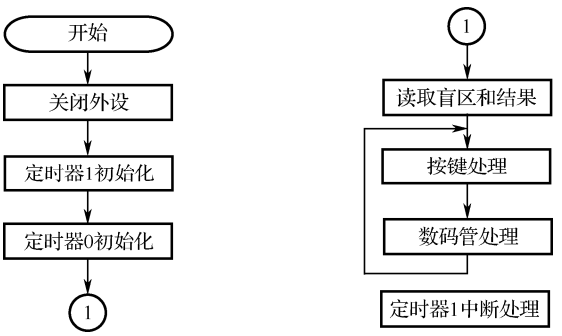



图 4.28　超声波测距机系统主程序流程图

超声波测距机“main.c”内容如下：

```
#include "tim.h"
#include "key.h"
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
#include "seg.h"
#include "stdio.h"
#include "iic.h"
#include "ultrasonic.h"

unsigned int  uiDac;
unsigned char pucDist[10], ucLast, ucTotal;
unsigned char ucNum_Data, ucNum_Echo, ucBlind;
unsigned char ucState, ucLed, ucOper_Flag, ucMeas_Num;
unsigned char pucSeg_Buf[9], pucSeg_Code[8], ucSeg_Pos;
unsigned char ucKey_Dly, ucKey_Old;
unsigned long ulms;

void Key_Proc(void);
void Seg_Proc(void);

void main(void)
{
  Cls_Peripheral();
  Timer1Init();
  Timer0Init();

  EEPROM_Read(pucSeg_Buf, 0, 1);
  if(pucSeg_Buf[0] <= 90) ucBlind = pucSeg_Buf[0];
  else ucBlind = 50;
  EEPROM_Read(pucDist, 1, 10);

  while(1)
  {
    Key_Proc();
    Seg_Proc();
  }
}

void Time_1(void) interrupt 3
{
  ulms++;
  if(++ucKey_Dly == 10)
    ucKey_Dly = 0;
  if(ucMeas_Num)
  {
    if(!(ulms % 200))
    {
      if(--ucMeas_Num) ucLed ^= 1;
      else ucLed &= 0xfe;
      Led_Disp(ucLed);
```



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
    }
  }
  Seg_Disp(pucSeg_Code, ucSeg_Pos);
  if(++ucSeg_Pos == 8) ucSeg_Pos = 0;
}

void Key_Proc(void)
{
  unsigned char ucKey_Val, ucKey_Down;

  if(ucKey_Dly) return;
  ucKey_Dly = 1;

  ucKey_Val = Key_Read();
  ucKey_Down = ucKey_Val & (ucKey_Old ^ ucKey_Val);
  ucKey_Old = ucKey_Val;

  switch(ucKey_Down)
  {
    case 4:                         // S4
      if(!ucState)
      {
        ucLast = pucDist[ucNum_Data];
        if(++ucNum_Data == 10) ucNum_Data = 0;
        pucDist[ucNum_Data] = Wave_Recv();
        ucTotal = ucLast + pucDist[ucNum_Data];

        if(pucDist[ucNum_Data] <= ucBlind)
          PCF8591_Dac(0);
        else
        {
          uiDac = (pucDist[ucNum_Data]-ucBlind)*1.02;     // 0.02*255/5
          if(uiDac > 255) uiDac = 255;
          PCF8591_Dac(uiDac);
        }

        pucSeg_Buf[0] = pucDist[ucNum_Data];
        EEPROM_Write(pucSeg_Buf, ucNum_Data+1, 1);
        ucMeas_Num = 20;
      }break;
    case 5:                         // S5
      if(!ucState)
      {
        ucState = 1;
        ucLed |= 0x40;
      }
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
      else if(ucState == 1)
      {
        ucState = 0;
        ucLed &= 0xbf;
      }break;
    case 9:                          // S9
      if(!ucState)
      {
        ucState = 2;
        ucLed |= 0x80;
      }
      else if(ucState == 2)
      {
        ucState = 0;
        ucLed &= 0x7f;
        pucSeg_Buf[0] = ucBlind;
        EEPROM_Write(pucSeg_Buf, 0, 1);
      }break;
    case 8:                          // S8
      switch(ucState)
      {
        case 0:
          ucOper_Flag ^= 1; break;
        case 1:
          if(++ucNum_Echo == 10) ucNum_Echo = 0; break;
        case 2:
          ucBlind += 10;
          if(ucBlind > 90) ucBlind = 0;
      }
  }
  Led_Disp(ucLed);
}

void Seg_Proc(void)
{
  switch(ucState)
  {
    case 0:
      if(!ucOper_Flag)
        sprintf(pucSeg_Buf, "0 %03u%03u", (unsigned int)ucLast,\
          (unsigned int)pucDist[ucNum_Data]);
      else
        sprintf(pucSeg_Buf, "1 %03u%03u", (unsigned int)ucTotal,\
          (unsigned int)pucDist[ucNum_Data]);
      break;
    case 1:
```



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
            sprintf(pucSeg_Buf, "%02u   %03u", (unsigned int)ucNum_Echo,\
              (unsigned int)pucDist[ucNum_Echo]);
            break;
          case 2:
            sprintf(pucSeg_Buf, "F    %02u", (unsigned int)ucBlind);
        }
        Seg_Tran(pucSeg_Buf, pucSeg_Code);
      }
```

### 4.4.2 系统实现

超声波测距机实现的主要步骤如下:

（1）在“D:\CT107D”文件夹中将“205_KEY”文件夹复制并重命名为“304_082”文件夹，进入“304_082”文件夹，双击工程文件“IAP15”打开工程。

（2）将“iic.c”和“ultrasonic.c”添加到“Source Group 1”中。

（3）修改“main.c”的内容。

（4）在“Options for Target”的“C51”标签中定义预处理符号“PCF8591_DAC EEPROM_AT24C02”。

（5）编译并调试程序。

**注意**：超声波测距时需要将 J2 的 1、3 和 2、4 短接。

### 4.4.3 客观题

（1）欲提高电压比较器的抗干扰能力，应选用下列哪种类型的比较器？（　　）

A．双限比较器　　B．滞回比较器

C．过零比较器　　D．单限比较器

（2）下列哪个 C51 关键字能够实现指定工作寄存器区？（　　）

A．interrupt　　B．code

C．using　　D．reentrant

（3）单个运算放大器和若干个电阻无法构成以下哪种电路？（　　）

A．比较器　　B．跟随器

C．乘法器　　D．振荡器

（4）关于 IAP15F2K61S2 单片机的中断错误的说法是（　　）。

A．上升沿和下降沿均可以触发外部中断请求

B．外部中断响应后，中断请求标注会自动清零，无须其他处理

C．EA 可以控制禁用所有中断源的中断请求

D．在中断源中断允许的条件下，单片机在任意时刻都能够响应中断请求

（5）超声波传感器基于下列哪个物理效应？（　　）

A．温度效应　　B．霍尔响应

C．压电效应　　D．横向效应

（6）某传感器输出电压信号（几乎不能够提供电流），经过放大后希望输出电压与信号成正比，此时应该选择（　　）放大电路。

A．电流串联负反馈　　B．电流并联负反馈

C．电压串联负反馈　　D．电压并联负反馈

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

（7）$N$ 个触发器构成的计数器中，有效状态最多有（　　）个。

A．$2^N$　　B．$N$

C．$2N-1$　　D．$2N$

（8）对结构体变量 sh 成员引用正确的是（　　）。

```
struct shape{
    float length;
    int width;
    int heigth;
}sh, *ptr;
ptr = &sh;
```

A．sh.width　　B．(*ptr).length

C．ptr→width　　D．ptr.length

（9）图 4.29 所示运放电路中电路的输出电压 $U_O$ 为_____V。

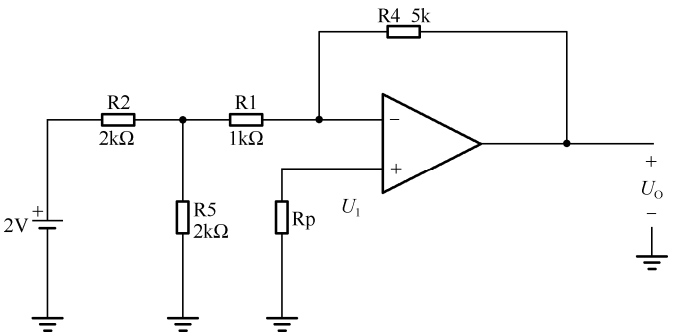



图 4.29　运放电路

（10）填空完成下列 C51 程序片段，以最合理的方式实现 MCS51 单片机将片外数据存储器中从 0x100 开始的 10 个字节数据传送到单片机片内 0x40 开始的区域内。

```
void transfer (void)
{
  unsigned char data index = 0;
  unsigned char * _________ mcu_ram;
  unsigned char * _________ extern_ram;
  mcu_ram = 0x40;
  extern_ram = 0x100;
  for(index=0; index<10; index++)
    *(mcu_ram + i) = *(extern_ram + i);
}
```

A．data　　B．pdata

C．idata　　D．xdata

参考答案与评分标准

（1）B（2 分）　（2）C（2 分）　（3）CD（5 分）　（4）D（2 分）　（5）C（2 分）

（6）C（2 分）　（7）A（2 分）　（8）D（2 分）　（9）-2.5（4 分）　（10）AC，D（7 分）



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

## 4.5 第九届预赛试题——彩灯控制器

彩灯控制器硬件框图如图 4.30 所示。

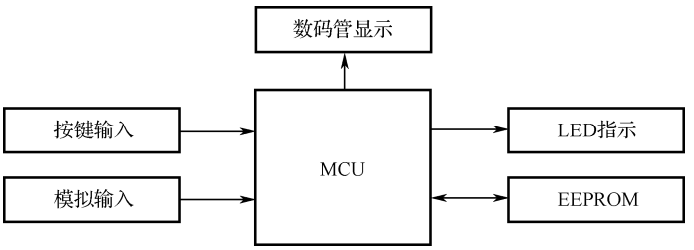



图 4.30　彩灯控制器硬件框图

彩灯控制器功能描述如下。

**1）基本功能描述**

通过单片机控制 8 个 LED 指示灯按照特定的顺序（工作模式）亮灭，指示灯的流转间隔可通过按键调整，亮度可由电位器 Rb2 进行控制，各工作模式的流转间隔时间需在 EEPROM 中保存，并可在硬件重新上电后自动载入。

**2）设计说明**

（1）关闭蜂鸣器、继电器等与本试题程序设计无关的外设资源。

（2）设备上电后默认数码管和 LED 指示灯均为熄灭状态。

（3）流转间隔可调整范围为 400～1200ms。

（4）设备固定按照模式 1、模式 2、模式 3 和模式 4 的次序循环往复运行。

**3）LED 指示灯工作模式**

（1）模式 1：按照 L1～L8 的顺序，从左到右单循环点亮。

（2）模式 2：按照 L8～L1 的顺序，从右到左单循环点亮。

（3）模式 3：按照 L1L8、L2L7、L3L6 和 L4L5 的顺序循环点亮。

（4）模式 4：按照 L4L5、L3L6、L2L7 和 L1L8 的顺序循环点亮。

**4）亮度等级控制**

检测电位器 Rb2 的输出电压，控制 8 个 LED 指示灯的亮度，要求在 0～5V 的可调区间内，实现 4 个均匀分布的 LED 指示灯亮度等级。

**5）按键功能（为了方便在线调试，对按键功能进行了调整）**

（1）按键 S5 定义为启动/停止按键，按下后启动或停止 LED 的流转。

（2）按键 S4 定义为设置按键，按下后数码管进入流转间隔设置界面，如图 4.31 所示。

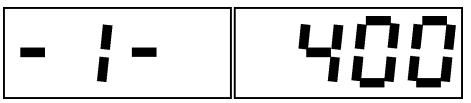



图 4.31　流转间隔设置界面

其中第 2 位为运行模式编号，后 3 位为流转间隔（单位：ms）。

通过按键 S4 可切换选择运行模式和流转间隔两个显示单元，当前被选择的显示单元以 0.8s 为间隔亮灭，设置完成后保存参数并退出设置界面。

（3）按键 S9 定义为加按键，在设置界面下，按下该键，若当前选择的是运行模式，则运行模式编号加 1，若当前选择的是流转间隔，则流转间隔增加 100ms。

（4）按键 S8 定义为减按键，在设置界面下，按下该键，若当前选择的是运行模式，则运行模

电子工业出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

式编号减 1，若当前选择的是流转间隔，则流转间隔减少 100ms。

（5）按键功能说明：

按键 S9 和 S8 的加减功能只在设置状态下有效，数值的调整应注意边界属性。

在非设置状态下，按下 S8 按键可显示 LED 当前的亮度等级，4 个亮度等级从暗到亮依次用 1、2、3、4 表示，松开 S8 按键，数码管显示关闭。亮度等级的显示格式如图 4.32 所示。

图 4.32　亮度等级显示格式

### 4.5.1　系统设计

彩灯控制器的详细框图如图 4.33 所示。

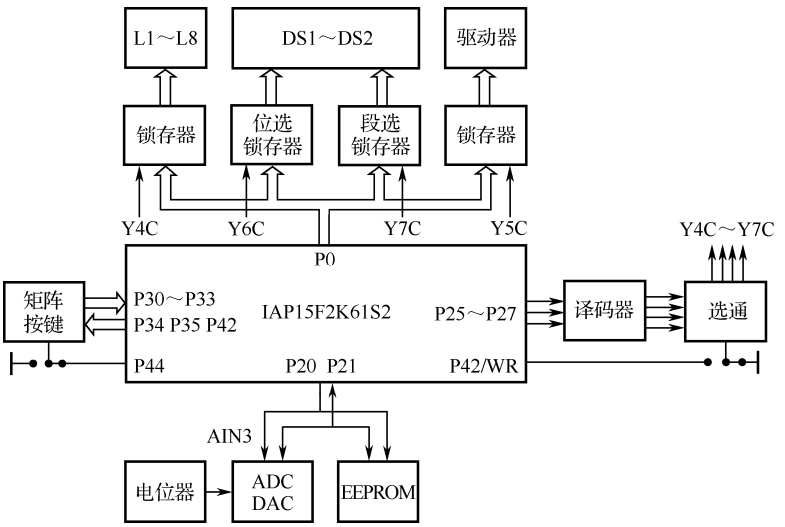



图 4.33　彩灯控制器详细框图

单片机采集模拟输入信号（电压信号通过 ADC 转换）和键盘功能要求，按要求设置 LED 工作模式、亮度和流转间隔，控制 LED 灯的点亮与熄灭，将相关数据显示在数码管上，24C02 作为 EEPROM 存储记录信息。

系统主程序流程图如图 4.34 所示。主程序首先关闭外设、对定时器 1 进行初始化，然后读取间隔并处理，最后进行按键处理、数码管处理和 LED 处理，其中按键处理中包含保存间隔，数码管处理中包含 PCF8591 ADC 处理。

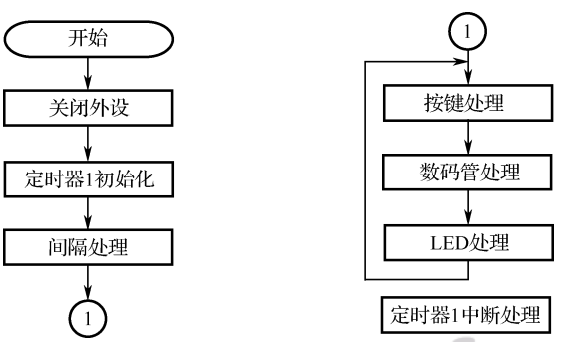



图 4.34　彩灯控制器主程序流程图



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

彩灯控制器“main.c”内容如下：

```
#include "tim.h"
#include "key.h"
#include "seg.h"
#include "stdio.h"
#include "iic.h"

unsigned int  puiInter[4];
unsigned char ucMode=1, ucLedi, ucLedj;
unsigned char ucBright=1, ucBright_Flag, ucRun, ucNum;
unsigned char pucLed_Val[4][8]
  ={0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80,
    0x80, 0x40, 0x20, 0x10, 0x08, 0x04, 0x02, 0x01,
    0x81, 0x42, 0x24, 0x18, 0x81, 0x42, 0x24, 0x18,
    0x18, 0x24, 0x42, 0x81, 0x18, 0x24, 0x42, 0x81};
unsigned char ucState, uc800ms, ucLed;
unsigned char pucSeg_Buf[9], pucSeg_Code[8], ucSeg_Pos;
unsigned char ucKey_Dly, ucKey_Old;
unsigned long ulms, ulStart;

void Int_Proc(void);
void Key_Proc(void);
void Seg_Proc(void);
void Led_Proc(void);

void main(void)
{
  Cls_Peripheral();
  Timer1Init();

  Int_Proc();

  while(1)
  {
    Key_Proc();
    Seg_Proc();
    Led_Proc();
  }
}

void Time_1(void) interrupt 3
{
  ulms++;
  if(++ucKey_Dly == 10)
    ucKey_Dly = 0;
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
  if(!(ulms % 800))
    uc800ms++;
  Seg_Disp(pucSeg_Code, ucSeg_Pos);
  if(++ucSeg_Pos == 8) ucSeg_Pos = 0;
  if(ucRun)
  {
    if(ucBright > ucNum)
      Led_Disp(ucLed);
    else
      Led_Disp(0);
    if(++ucNum == 5)
      ucNum = 0;
  }
  else
    Led_Disp(0);
}

void Int_Proc(void)
{
  unsigned char i;
  EEPROM_Read(pucSeg_Buf, 0, 4);
  for(i=0; i<4; i++)
    if((puiInter[0]>=4) && (puiInter[0]<=12))
      puiInter[i] = pucSeg_Buf[i]*100;
    else
      puiInter[i] = 400;
}

void Key_Proc(void)
{
  unsigned char ucKey_Val, ucKey_Down;

  if(ucKey_Dly) return;
  ucKey_Dly = 1;

  ucKey_Val = Key_Read();
  ucKey_Down = ucKey_Val & (ucKey_Old ^ ucKey_Val);
  ucKey_Old = ucKey_Val;

  switch(ucKey_Down)
  {
    case 5:                          // S5
      ucRun ^= 1;
      ulStart = ulms;
      ucLed = pucLed_Val[0][0];
      ucLedi = 0;
```



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
      ucLedj = 1;
      break;
    case 4:                          // S4
      if(++ucState == 3)
      {
        ucState = 0;
        pucSeg_Buf[0] = puiInter[0]/100;
        pucSeg_Buf[1] = puiInter[1]/100;
        pucSeg_Buf[2] = puiInter[2]/100;
        pucSeg_Buf[3] = puiInter[3]/100;
        EEPROM_Write(pucSeg_Buf, 0, 4);
      }break;
    case 9:
      if(ucState == 1)
        if(++ucMode == 5)
          ucMode = 1;
      if(ucState == 2)
      {
        puiInter[ucMode-1] += 100;
        if(puiInter[ucMode-1] == 1300)
          puiInter[ucMode-1] = 400;
      }break;
    case 8:
      if(ucState == 1)
        if(--ucMode == 0)
          ucMode = 4;
      if(ucState == 2)
      {
        puiInter[ucMode-1] -= 100;
        if(puiInter[ucMode-1] == 300)
          puiInter[ucMode-1] = 1200;
      }
  }
  if(!ucState && (ucKey_Old == 8))
    ucBright_Flag = 1;
  if(!ucState && !ucKey_Old)
    ucBright_Flag = 0;
}

void Seg_Proc(void)
{
  if(ucState)
  {
    sprintf(pucSeg_Buf, "-%1u- %4u", (unsigned int)ucMode,\
      (unsigned int)puiInter[ucMode-1]);
    if(uc800ms&1)
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
    {
      if(ucState == 1)
        pucSeg_Buf[1] = 0xff;
      else
      {
        pucSeg_Buf[4] = 0xff;
        pucSeg_Buf[5] = 0xff;
        pucSeg_Buf[6] = 0xff;
        pucSeg_Buf[7] = 0xff;
      }
    }
  }
  else
  {
    if(ucBright_Flag)
      sprintf(pucSeg_Buf, "     -%1u", (unsigned int)ucBright);
    else
      sprintf(pucSeg_Buf, "       ");
  }
  Seg_Tran(pucSeg_Buf, pucSeg_Code);
  ucBright = (PCF8591_Adc()>>6)+1;
}

void Led_Proc(void)
{
  if(ulms-ulStart >= puiInter[ucLedi])
  {
    ulStart = ulms;
    ucLed = pucLed_Val[ucLedi][ucLedj];
    if(++ucLedj == 8)
    {
      ucLedj = 0;
      if(++ucLedi == 4)
        ucLedi = 0;
    }
  }
}
```

### 4.5.2 系统实现

彩灯控制器实现的主要步骤如下:

（1）在“D:\CT107D”文件夹中将“205_KEY”文件夹复制并重命名为“305_091”文件夹，进入“305_091”文件夹，双击工程文件“IAP15”打开工程。

（2）将“iic.c”添加到“Source Group 1”中。

（3）修改“main.c”的内容。

（4）在“Options for Target”的“C51”标签中定义预处理符号“PCF8591_ADC EEPROM_



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

AT24C02”。

（5）编译并调试程序。

### 4.5.3 客观题

（1）当MCS-51访问片外的存储器时，其低8位地址由_____口提供，高8位地址由_____口提供，8位数据由_____口提供。

（2）当由MCS-51单片机构成的系统正常工作后，在RST引脚上附加一个_____电平，并至少维持_____个机器周期可令系统复位，复位后各IO口为_____电平。

（3）当温度升高时，二极管的反向饱和电流将（　　）。

A．增大　　B．减小

C．保持不变　　D．与温度没有直接关系

（4）下列哪个C51关键字能够将数据存储在程序存储区中？（　　）

A．xdata　　B．idata

C．bdata　　D．code

（5）设计一位8421 BCD码计数器至少需要（　　）个触发器。

A．3　　B．4

C．5　　D．8

（6）如图4.35所示共阴极数码管显示“F”的编码是（　　）。

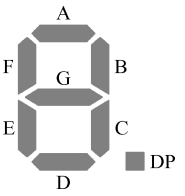



图4.35　共阴极数码管

A．0xC8　　B．0x71

C．0xD9　　D．0xE2

（7）为了使高阻信号源与低阻负载进行配合，在设计电路过程中往往需要进行阻抗匹配，以下哪种电路适合接入高阻信号源与低阻负载之间？（　　）

A．共射电路　　B．共基电路

C．共集电路　　D．以上都可以

（8）在C51中以下哪种数据类型能够表达的数值最大？（　　）

A．char　　B．long

C．int　　D．float

（9）运放电路如图4.36所示，输入与输出之间的关系是（　　）。

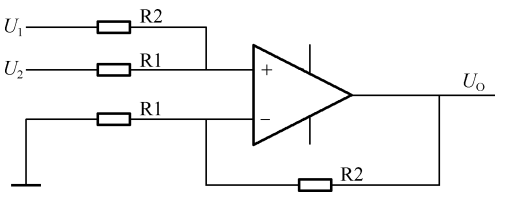



图4.36　运放电路

A．$U_O = U_1 + U_2 \times R1/R2$　　B．$U_O = U_1 + U_2 \times R2/R1$

电子工业出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

C．$U_O = U_1 \times R2/R1 + U_2$　　D．$U_O = U_1 \times R1/R2 + U_2$

（10）关于 MCS-51 单片机，以下说法中错误的有（　　）。

A．单片机数据存储器和程序存储器扩展的最大范围是一样的。

B．串口数据发送和接收缓冲器均为 SBUF，不能够同时发送和接收数据。

C．为消除按键产生的抖动，可以采用软件和硬件两种办法。

D．单片机上电复位后，片内数据存储器的内容均为 00H。

参考答案与评分标准。

（1）P0、P2、P0（3 分，每空 1 分）

（2）高，2，高（3 分，每空 1 分）

（3）A（3 分）　（4）D（3 分）　（5）B（3 分）　（6）BD（3 分）

（7）C（3 分）　（8）D（3 分）　（9）B（3 分）　（10）BD（3 分）

## 4.6　第九届决赛试题——多功能测量仪表

多功能测量仪表硬件框图如图 4.37 所示。

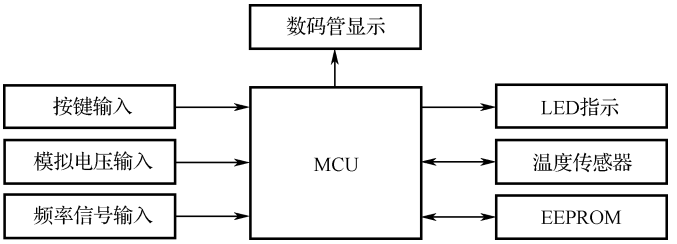



图 4.37　多功能测量仪表硬件框图

多功能测量仪表功能描述如下：

**1）基本功能**

（1）测量实训平台上 Rb2 输出的电压信号和 NE555 模块输出的频率信号，采集 DS18B20 温度传感器输出的温度数据，通过数码管显示，并可随时存储。

（2）通过实训平台上的按键可以完成切换数码管显示内容、保存数据、回显数据等功能。

（3）可实现电压阈值设置功能和报警提醒功能。

（4）频率测量功能需将实训平台 J3-SIGNAL 引脚与 P34 引脚短接（SIGNAL 与 P34 的短接可以使用超声/红外切换等与本试题功能要求无关的跳线帽完成）。

**2）显示功能**

（1）数据显示界面：包括电压显示、频率显示和温度显示等。电压显示界面如图 4.38 所示，显示内容包括提示符 U 和电压值，电压测量结果保留小数点后 1 位有效数字，单位为 V。

图 4.38　电压显示界面

频率显示界面如图 4.39 所示，显示内容包括提示符 F 和频率值（后 6 位，未使用的位应熄灭），频率的单位为 Hz。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

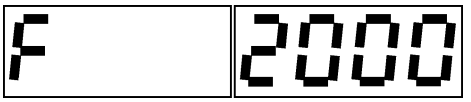



图 4.39　频率显示界面

温度显示界面如图 4.40 所示，显示内容包括提示符 C 和温度值，温度测量结果保留小数点后 2 位有效数字，单位为℃。

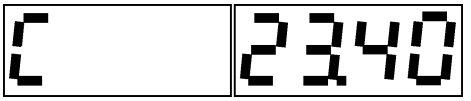



图 4.40　温度显示界面

（2）数据回显界面：数据回显界面如图 4.41 所示，显示内容包括回显提示符 H、回显数据提示符（电压：U、频率：F 或温度：C）和回显数据（后 6 位，未使用的位应熄灭）。

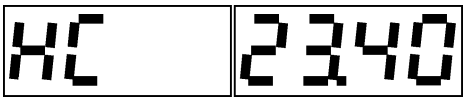



图 4.41　数据回显界面

（3）电压阈值设置界面：电压阈值设置界面如图 4.42 所示，显示内容包括参数提示符 P 和电压阈值参数。

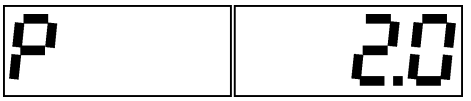



图 4.42　电压阈值设置界面

**3）按键功能（为了方便在线调试，对按键功能进行了调整）**

S4：定义为切换按键，按下此按键，切换数码管的显示状态（电压、频率和温度)，切换按键在数据显示界面和数据回显界面下均有效。

S5：定义为存储按键，按下此按键，将当前设备测量到的电压、频率和温度数据同时保存到 EEPROM 中。

S8：定义为回显按键，按下此按键，进入数据回显界面，可显示最近一次保存在 EEPROM 中的电压、频率和温度数据。

S9：定义为参数设置按键，按下此按键，进入电压阈值参数设置界面。在阈值设置界面下，S8 定义为阈值调整功能，每次按下 S8，电压阈值增加 0.1V，长按 0.8s 以上，可实现快速增加功能（电压阈值设置范围为 0.1～5.0V），阈值调整完成后，再次按下 S9，退出设置界面，返回数据显示界面。

**4）存储功能**

通过 EEPROM 可实现电压、频率、温度数据记录和电压阈值参数的存储功能，设备重新上电后，能够自动从 EEPROM 中载入全部参数。

**5）LED 指示灯功能**

（1）电压测量状态下 L1 点亮。

（2）频率测量状态下 L2 点亮。

（3）温度测量状态下 L3 点亮。

（4）当 Rb2 输出电压大于电压阈值时，L8 以 0.2s 为间隔闪烁。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

### 4.6.1 系统设计

多功能测量仪表详细框图如图 4.43 所示。电压信号通过 ADC 转换，单片机采集该模拟输入信号的 ADC 数值，555 方波输出至单片机 P34 引脚，作为计数器 0 输入端计数便于方波频率计算，DS18B20 为温度传感器，单片机从 DS18B20 采集温度信息、从键盘采集功能要求，并将电压、频率和温度等数据按要求显示到数码管，控制 LED 灯点亮与熄灭，24C02 作为 EEPROM 存储记录数据。

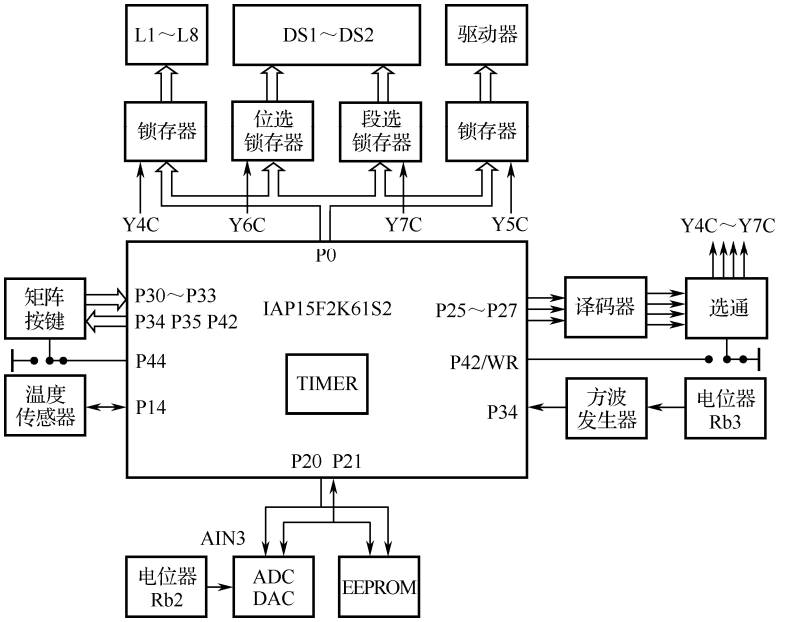



图 4.43　多功能测量仪表详细框图

系统主程序流程图如图 4.44 所示。主程序首先关闭外设，对定时器 1 和定时器 0 进行初始化并读取存储数据，然后进行按键处理和数码管处理，其中按键处理中包含存储数据，数码管处理中包含电压和温度测量，频率测量在定时器 1 中断服务程序中实现。

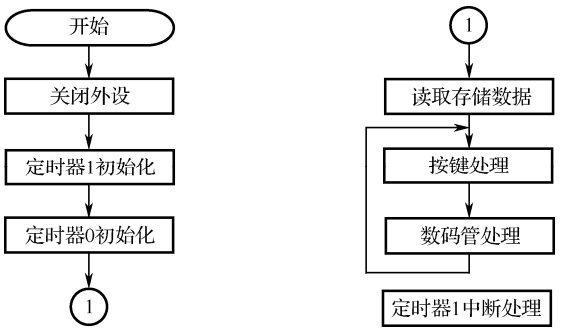



图 4.44　多功能测量仪表主程序流程图

多功能测量仪表“main.c”内容如下：

```
#include "tim.h"
#include "key.h"
#include "seg.h"
#include "stdio.h"
#include "onewire.h"
```



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
#include "iic.h"

unsigned int  uiFre_Val, uiTem_Val, uiSeg_Dly;
unsigned int  uiFre_Echo, uiTem_Echo;
unsigned char ucVol_Val, ucVol_Echo, ucVol_Thr, ucVol_Flag;
unsigned char ucState, ucLed=1;
unsigned char pucSeg_Buf[10], pucSeg_Code[8], ucSeg_Pos;
unsigned char ucKey_Dly, ucKey_Old;
unsigned long ulms, ulKey_Time;

void Timer0Init(void)                   // 16 位计数器模式
{
  AUXR &= 0x7F;                         // 定时器时钟 12T 模式
  TMOD |= 5;                            // 设置 16 位计数器模式
  TL0 = 0;                              // 设置计数初值
  TH0 = 0;                              // 设置计数初值
  TF0 = 0;                              // 清除 TF0 标志
  TR0 = 1;                              // 开始计数
}

void Key_Proc(void);
void Seg_Proc(void);

void main(void)
{
  Cls_Peripheral();
  Timer1Init();
  Timer0Init();

  EEPROM_Read(pucSeg_Buf, 0, 6);
  ucVol_Val = pucSeg_Buf[0];
  uiFre_Val = (pucSeg_Buf[2]<<8)+pucSeg_Buf[1];
  uiTem_Val = (pucSeg_Buf[4]<<8)+pucSeg_Buf[3];
  ucVol_Thr = pucSeg_Buf[5];
  if(ucVol_Thr > 50) ucVol_Thr = 20;

  while(1)
  {
    Key_Proc();
    Seg_Proc();
  }
}

void Time_1(void) interrupt 3
{
  ulms++;
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
  if(++ucKey_Dly == 10)
    ucKey_Dly = 0;
  if(++uiSeg_Dly == 500)
    uiSeg_Dly = 0;
  if(ucVol_Flag)
    if(!(ulms % 200))
      ucLed ^= 0x80;
  if(!(ulms % 1000))
  {
    uiFre_Val = (TH0<<8)+TL0;
    TH0 = 0;
    TL0 = 0;
  }
  Led_Disp(ucLed);
  Seg_Disp(pucSeg_Code, ucSeg_Pos);
  if(++ucSeg_Pos == 8) ucSeg_Pos = 0;
}

void Key_Proc(void)
{
  unsigned char ucKey_Val, ucKey_Down;

  if(ucKey_Dly) return;
  ucKey_Dly = 1;

  ucKey_Val = Key_Read();
  ucKey_Down = ucKey_Val & (ucKey_Old ^ ucKey_Val);
  ucKey_Old = ucKey_Val;
  if(ucKey_Down) ulKey_Time = ulms;

  switch(ucKey_Down)
  {
    case 4:                          // S4
      if(ucState < 3)
      {
        if(++ucState == 3) ucState = 0;
        if(!ucState) ucLed = 1;
        if(ucState == 1) ucLed = 2;
        if(ucState == 2) ucLed = 4;
      }
      else if(ucState < 0x13)
        if(++ucState == 0x13) ucState = 0x10;
      break;
    case 5:                          // S5
      pucSeg_Buf[0] = ucVol_Val;
      pucSeg_Buf[1] = uiFre_Val & 0xff;
```



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
      pucSeg_Buf[2] = uiFre_Val >> 8;
      pucSeg_Buf[3] = uiTem_Val & 0xff;
      pucSeg_Buf[4] = uiTem_Val >> 8;
      EEPROM_Write(pucSeg_Buf, 0, 5);
      break;
    case 8:                           // S8
      if(ucState < 0x10)
      {
        ucState = 0x10;
        ucLed = 0;
        EEPROM_Read(pucSeg_Buf, 0, 5);
        ucVol_Echo = pucSeg_Buf[0];
        uiFre_Echo = (pucSeg_Buf[2]<<8)+pucSeg_Buf[1];
        uiTem_Echo = (pucSeg_Buf[4]<<8)+pucSeg_Buf[3];
      }
      else if(ucState < 0x20)
      {
        ucState = 0;
        ucLed = 1;
      }
      if(ucState == 0x20)
        if(++ucVol_Thr > 50) ucVol_Thr = 1;
      break;
    case 9:                           // S9
      if(ucState != 0x20) ucState = 0x20;
      else
      {
        ucState = 0;
        pucSeg_Buf[0] = ucVol_Thr;
        EEPROM_Write(pucSeg_Buf, 5, 1);
      }
  }

  if((ucState==0x20) && (ucKey_Old==8))
    if(ulms-ulKey_Time > 800)
    {
      ulKey_Time = ulms;
      if(++ucVol_Thr > 50) ucVol_Thr = 1;
    }
}

void Seg_Proc(void)
{
  if(uiSeg_Dly) return;
  uiSeg_Dly = 1;
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
    switch(ucState&0xf0)
    {
      case 0:                           // 测量
        switch(ucState)
        {
          case 0:                       // 电压
            ucVol_Val = PCF8591_Adc()*50/255;
            if(ucVol_Val > ucVol_Thr)
              ucVol_Flag = 1;
            else
            {
              ucVol_Flag = 0;
              ucLed &= 0x7f;
            }
            sprintf(pucSeg_Buf, "U    %02.1f", (float)ucVol_Val/10.0);
            break;
          case 1:                       // 频率
            sprintf(pucSeg_Buf, "F  %5u", (unsigned int)uiFre_Val);
            break;
          case 2:                       // 温度
            uiTem_Val = rd_temperature();
            sprintf(pucSeg_Buf, "C   %04.2f", (float)uiTem_Val/16.0);
        }break;
      case 0x10:                        // 回显
        switch(ucState)
        {
          case 0x10:                    // 电压
            sprintf(pucSeg_Buf, "HU   %02.1f", (float)ucVol_Echo/10.0);
            break;
          case 0x11:                    // 频率
            sprintf(pucSeg_Buf, "HF %5u", uiFre_Echo);
            break;
          case 0x12:                    // 温度
            sprintf(pucSeg_Buf, "HC  %04.2f", (float)uiTem_Echo/16.0);
        }break;
      case 0x20:
        sprintf(pucSeg_Buf, "P    %02.1f", (float)ucVol_Thr/10.0);
    }
    Seg_Tran(pucSeg_Buf, pucSeg_Code);
  }
```

### 4.6.2 系统实现

多功能测量仪表实现的主要步骤如下:

（1）在“D:\CT107D”文件夹中将“208_DS18B20”文件夹复制并重命名为“306_092”文件夹，进入“306_092”文件夹，双击工程文件“IAP15”打开工程。



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

（2）将“iic.c”添加到“Source Group 1”中。

（3）修改“main.c”的内容。

（4）在“Options for Target”的“C51”标签中定义预处理符号“PCF8591_ADC EEPROM_AT24C02”。

（5）编译并调试程序。

**注意：**频率测量时需要将 J3-15（SIGNAL）与 J3-16（P34）短接。

### 4.6.3 客观题

（1）IAP15F2K61S2 单片机具有_____个串口，单片机内部_____个外设可作为串口 1 波特率发生器。

（2）由理想运算放大器构成的小信号交流放大电路如图 4.45 所示，频带内的电压放大倍数为_____，下限频率 $f_L$ 为_____Hz（电压放大倍数和截止频率均取整数部分）。

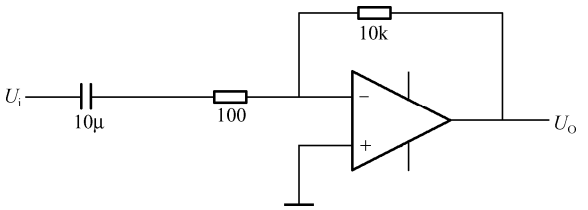



图 4.45　小信号交流放大电路

（3）单片机内部反应程序运行状态或运算结果的特征寄存器是（　　）。

A．PC　　B．PSW

C．SP　　D．A

（4）以下哪个选项不是 C51 的基本数据类型？（　　）

A．char　　B．int

C．void　　D．double

（5）测得某些电路中三极管的各极电压如图 4.46 所示，哪些三极管处于放大状态？（　　）

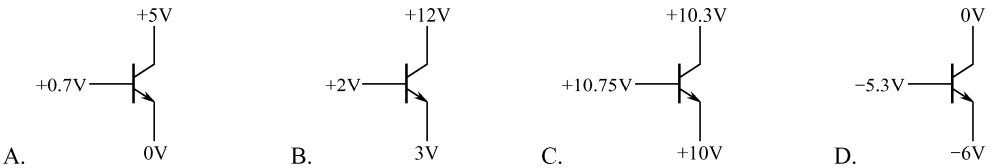



图 4.46　小信号交流放大电路

（6）同步时序电路和异步时序电路比较，其差异在于后者（　　）。

A．没有触发器　　B．没有统一的时钟控制脉冲

C．没有稳定状态　　D．输出只与内部状态有关

（7）C51 中一般指针变量需要占据几个字节的存储空间？（　　）

A．1　　B．2

C．3　　D．4

（8）以下表达式中符合逻辑运算法则的是（　　）。

A．C·C = C2　　B．1 + 1 = 10

C．0 < 1　　D．C + 1 = 1

（9）以下关于单片机的说法中错误的是（　　）。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

A．堆栈是单片机内部的一个特殊区域，与 RAM 无关

B．汇编语言指令是指能够被单片机直接执行的指令

C．单片机的总线宽度决定了单片机的寻址能力

D．单片机对内部 RAM 和外部 RAM 的读写速度是不同的

（10）两个电压放大倍数相同（电路相同，且采用同一种晶体管）的电路 A 和 B，对同一个信号源的电压进行放大，在负载开路的条件下，测得 A 电路的输出电压较小，不考虑仪表的测量误差，这说明 A 电路（　　）。

A．输入电阻大　　B．输出电阻大

C．输入电阻小　　D．输出电阻小

参考答案与评分标准

（1）2，2（3 分），每空 1.5 分；（2）-100，159（3 分），每空 1.5 分）

（3）B（3 分）　（4）D（3 分）　（5）AD（3 分）　（6）B（3 分）

（7）C（3 分）　（8）D（3 分）　（9）AB（3 分）　（10）C（3 分）

## 4.7　第十届预赛试题

系统硬件框图如图 4.47 所示。

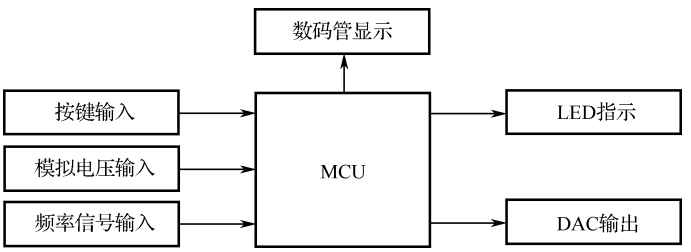



图 4.47　系统硬件框图

系统功能描述如下。

**1）基本功能**

（1）测量实训平台上 Rb2 输出的电压信号和 NE555 模块输出的频率信号，以数码管和 LED 等外围设备进行数据呈现。

（2）频率测量功能需将实训平台的 J3-SIGNAL 引脚与 P34 引脚短接（SIGNAL 与 P34 的短接可以使用超声/红外切换等与本试题功能要求无关的跳线帽完成）。

（3）使用 PCF8591 测量电位器 Rb2 的输出电压，并根据试题要求通过其 DAC 功能输出该电压值（J3-D/A）。

（4）电压、频率数据刷新时间要求：

- 电压数据刷新时间≤0.5s。
- 频率数据刷新时间≤1s。

（5）电压、频率数据测量范围要求。

- 电压数据测量范围：电位器 Rb2 输出的最小电压值到最大电压值。
- 频率数据测量范围：NE555 模块输出的最低频率到最高频率值。

**2）显示功能**

（1）电压显示界面：电压显示界面如图 4.48 所示，显示内容包括提示符 U 和电位器 Rb2 输



电子工业出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

出的电压值，电压测量结果保留小数点后 2 位有效数字，单位为 V。

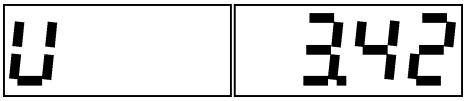



图 4.48　电压显示界面

（2）频率显示界面：频率显示界面如图 4.49 所示，显示内容包括提示符 F 和频率值，频率的单位为 Hz。

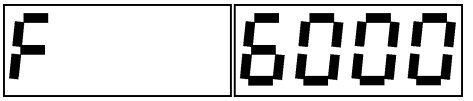



图 4.49　频率显示界面

**注**：频率数据显示使用 6 位数码管，当显示的数据长度不足 6 位时，未使用到的数码管位应熄灭。

**3）按键功能（为了方便在线调试，对按键功能进行了调整）**

S4：定义为显示界面切换按键，按下此按键，切换电压显示界面和频率显示界面。

S5：定义为 DAC 输出模式切换按键，按下此按键，DAC 输出模式在固定输出电压 2V 和跟随输出 Rb2 电压间切换。

S8：定义为 LED 功能控制按键，按下此按键，关闭或打开 LED 指示功能。

S9：定义为数码管功能控制按键，按下此按键，关闭或打开数码管显示功能。

**4）LED 指示功能**

（1）电压测量功能指示：L1 点亮，L2 熄灭。

（2）频率测量功能指示：L1 熄灭，L2 点亮。

（3）L3 功能：

- $V_{Rb2}$<1.5V 或 2.5V≤$V_{Rb2}$<3.5V 时 L3 熄灭。
- 1.5V≤$V_{Rb2}$<2.5V 或 $V_{Rb2}$≥3.5V 时 L3 点亮。

（4）L4 功能：

- $F_{OUT}$<1kHz 或 5kHz≤$F_{OUT}$<10kHz 时 L4 熄灭。
- 1kHz≤$F_{OUT}$<5kHz 或 $F_{OUT}$≥10kHz 时 L4 点亮。

（5）L5 功能：

- DAC 输出固定电压（2.0V）时，L5 熄灭。
- DAC 输出电压跟随 Rb2 电位器输出电压变化时，L5 点亮。

（6）本试题未涉及的 LED 应处于熄灭状态。

**5）初始状态说明**

（1）初始状态上电默认处于电压测量状态，数码管显示和 LED 指示功能启用。

（2）初始状态上电默认 PCF8591 DAC 芯片输出固定电压值 2.0V。

### 4.7.1　系统设计

系统详细框图如图 4.50 所示。单片机通过 PCF8591 ADC 采集 Rb2 的电压并通过 DAC 输出电压，555 方波输出至单片机 P34 引脚，通过计数器 0 计数得到方波频率，从键盘采集功能要求，并将电压和频率等数据按要求显示到数码管，控制 LED 的点亮与熄灭。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

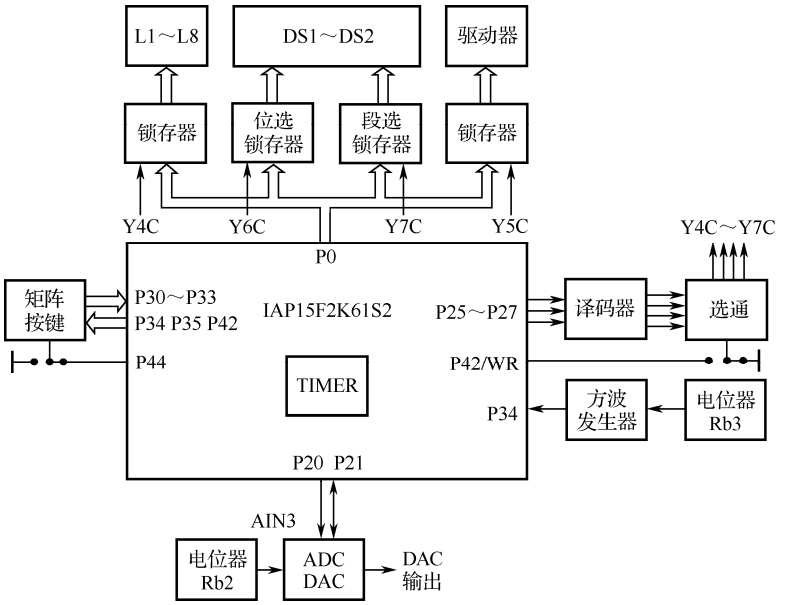



图 4.50　系统详细框图

系统主程序流程图如图 4.51 所示。主程序首先关闭外设，对定时器 1 和定时器 0 进行初始化，然后进行按键处理、数码管处理和 LED 处理，其中数码管处理中包含 ADC 输入和 DAC 输出，频率测量在定时器 1 中断服务程序中实现。

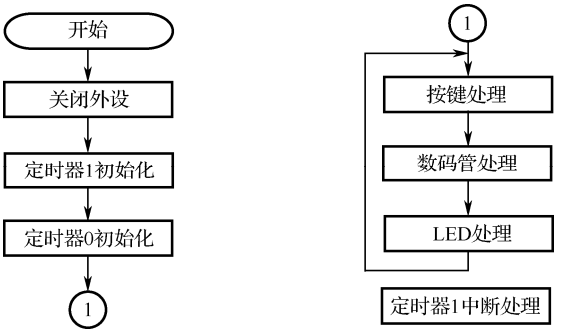



图 4.51　系统主程序流程图

系统“main.c”内容如下：

```
#include "tim.h"
#include "key.h"
#include "seg.h"
#include "stdio.h"
#include "iic.h"

unsigned int  uiFre_Val, uiVol_Val, uiSeg_Dly;
unsigned char ucAdc_Val, ucLed_Ctrl=1, ucSeg_Ctrl=1;
unsigned char ucState, ucLed, ucDac_Flag=1;
unsigned char pucSeg_Buf[10], pucSeg_Code[8], ucSeg_Pos;
unsigned char ucKey_Dly, ucKey_Old;
unsigned long ulms;
oid Led_Proc(void);
```



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
void Timer0Init(void)                   // 16 位计数器模式
{
  AUXR &= 0x7F;                         // 定时器时钟 12T 模式
  TMOD |= 5;                            // 设置 16 位计数器模式
  TL0 = 0;                              // 设置计数初值
  TH0 = 0;                              // 设置计数初值
  TF0 = 0;                              // 清除 TF0 标志
  TR0 = 1;                              // 开始计数
}

void Key_Proc(void);
void Seg_Proc(void);
v
void main(void)
{
  Cls_Peripheral();
  Timer1Init();
  Timer0Init();

  while(1)
  {
    Key_Proc();
    Seg_Proc();
    Led_Proc();
  }
}

void Time_1(void) interrupt 3
{
  ulms++;
  if(++ucKey_Dly == 10) ucKey_Dly = 0;
  if(++uiSeg_Dly == 500) uiSeg_Dly = 0;
  if(!(ulms % 1000))
  {
    uiFre_Val = (TH0<<8)+TL0;
    TH0 = 0;
    TL0 = 0;
  }
  Led_Disp(ucLed);
  Seg_Disp(pucSeg_Code, ucSeg_Pos);
  if(++ucSeg_Pos == 8) ucSeg_Pos = 0;
}

void Key_Proc(void)
{
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
  unsigned char ucKey_Val, ucKey_Down;

  if(ucKey_Dly) return;
  ucKey_Dly = 1;

  ucKey_Val = Key_Read();
  ucKey_Down = ucKey_Val & (ucKey_Old ^ ucKey_Val);
  ucKey_Old = ucKey_Val;

  switch(ucKey_Down)
  {
    case 4:                          // S4
      ucState ^= 1; break;
    case 5:                          // S5
      ucDac_Flag ^= 1; break;
    case 8:                          // S8
      ucLed_Ctrl ^= 1; break;
    case 9:                          // S9
      ucSeg_Ctrl ^= 1;
  }
}

void Seg_Proc(void)
{
  if(uiSeg_Dly) return;
  uiSeg_Dly = 1;

  ucAdc_Val = PCF8591_Adc();
  if(ucDac_Flag) PCF8591_Dac(102);                                  // 255*2/5
  else PCF8591_Dac(ucAdc_Val);

  if(ucSeg_Ctrl)
  {
    if(!ucState)
      sprintf(pucSeg_Buf, "U   %03.2f", (float)ucAdc_Val/51.0);  // 5/255
    else
      sprintf(pucSeg_Buf, "F  %5u", (unsigned int)uiFre_Val);
  }
  else
    sprintf(pucSeg_Buf, "       ");
  Seg_Tran(pucSeg_Buf, pucSeg_Code);
}

void Led_Proc(void)
{
  if(ucLed_Ctrl)
```



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
    {
      if(!ucState)
      {
        ucLed |= 1; ucLed &= ~2;
      }
      else
      {
        ucLed |= 2; ucLed &= ~1;
      }

      uiVol_Val = ucAdc_Val*100/51;                                    // 500/255
      if((uiVol_Val < 150) || ((uiVol_Val >= 250) && (uiVol_Val < 350)))
        ucLed &= ~4;
      else
        ucLed |= 4;

      if((uiFre_Val < 1000) || ((uiFre_Val >= 5000) && (uiFre_Val < 10000)))
        ucLed &= ~8;
      else
        ucLed |= 8;

      if(ucDac_Flag) ucLed &= ~0x10;
      else ucLed |= 0x10;
    }
    else
      ucLed = 0;
  }
```

### 4.7.2 系统实现

系统实现的主要步骤如下:

（1）在“D:\CT107D”文件夹中将“205_KEY”文件夹复制并重命名为“307_101”文件夹，进入“307_101”文件夹，双击工程文件“IAP15”打开工程。

（2）将“iic.c”添加到“Source Group 1”中。

（3）修改“main.c”的内容。

（4）在“Options for Target”的“C51”标签中定义预处理符号“PCF8591_ADC PCF8591_ DAC”。

（5）编译并调试程序。

**注意：**频率测量时需要将 J3-15（SIGNAL）与 J3-16（P34）短接。

### 4.7.3 客观题

（1）二极管电路如图 4.52 所示，若二极管的导通电压为 0.7V，输出电压 $U_O$ 为（　　）。

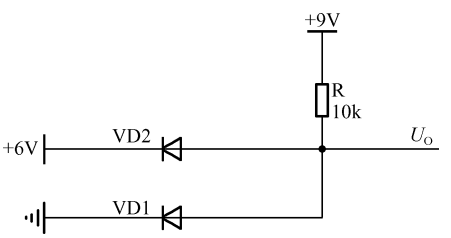


图 4.52　二极管电路

A．9V　　B．0.7V

C．6.7V　　D．0.35V

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

（2）当 MCS-51 单片机扩展外部存储器时，P2 口可作为（　　）。

A．8 位数据输入口　　B．8 位数据输出口

C．输出高 8 位地址　　D．输出低 8 位地址

（3）IAP15F2K61S2 单片机内部有（　　）个定时/计数器，工作模式最少的是（　　）。

A．3，定时器 0　　B．3，定时器 2

C．4，定时器 1　　D．4，定时器 2

（4）某存储器芯片的地址线为 12 根，数据线 16 根，它的存储容量为（　　）。

A．1KB　　B．2KB

C．4KB　　D．8KB

（5）将三角波转换为矩形波，需选用（　　）。

A．多谐振荡器　　B．双稳态触发器

C．单稳态触发器　　D．施密特触发器

（6）在 IAP15F2K61S2 单片机中，下列寄存器与定时器工作模式配置无关的是（　　）。

A．AUXR　　B．SCON

C．TCON　　D．PCON

（7）放大电路在负载开路时的输出电压为 0.4V，接入 3kΩ 的电阻负载后，输出的电压降为 0.3V，则该放大电路的输出电阻为（　　）。

A．10kΩ　　B．2kΩ

C．3kΩ　　D．1kΩ

（8）某放大电路中使用三极管的极限参数为 $P_{CM}$ = 100mW，$I_{CM}$ = 20mA，$U_{(BR)CEO}$ = 15V。以下哪些情况下，三极管不能正常工作？（　　）

A．$U_{CE}$ = 3V，$I_C$ = 15mA　　B．$U_{CE}$ = 2V，$I_C$ = 40mA

C．$U_{CE}$ = 6V，$I_C$ = 20mA　　D．$U_{CE}$ = 9V，$I_C$ = 10mA

（9）运放电路如图 4.53 所示，输入电压 $U_1$=0.4V，$U_2$=0.8V，输出电压 $U$o 的值为（　　）。

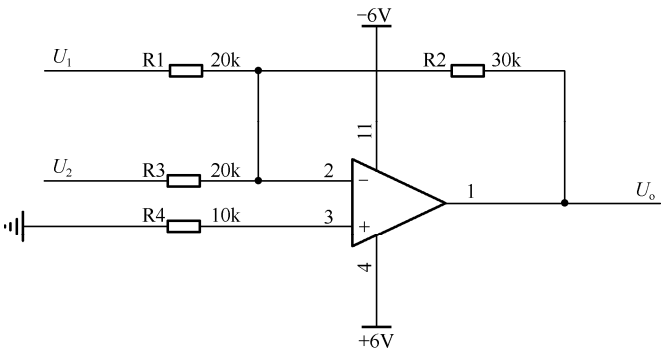



图 4.53　运放电路

A．1.2V　　B．−1.2V

C．−6V　　D．−1.8V

（10）下列说法中正确的是（　　）。

A．IAP15F2K61S2 单片机可以通过串口实现在线仿真功能。

B．单片机竞赛实训平台在 IO 和 MM 模式下，均可实现对数码管和 LED 指示灯的分别操作，互不影响。

C．对 DS1302 进行单字节写操作时，数据在时钟线 SCLK 下降沿写入 DS1302。



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

D．I2C 总线的启动信号和停止信号，只能由主器件发起。

参考答案与评分标准

（1）B（3 分）　（2）C（3 分）　（3）B（3 分）　（4）D（3 分）　（5）D（3 分）

（6）BD（3 分）　（7）D（3 分）　（8）BC（3 分）　（9）D（3 分）　（10）ABD（3 分）

## 4.8　第十届决赛试题

系统硬件框图如图 4.54 所示。

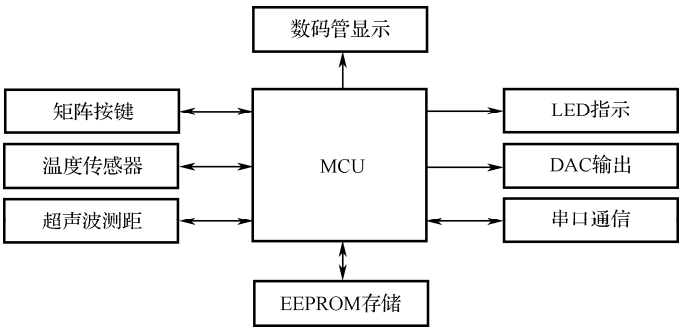



图 4.54　系统硬件框图

系统功能描述如下：

**1）基本功能**

（1）使用超声波测距单元完成测距功能。

（2）通过 DS18B20 温度传感器完成温度测量功能。

（3）通过 PCF8591 DAC 完成模拟电压输出功能。

（4）通过 AT24C02 EEPROM 存储器完成参数变动次数记录功能。

（5）通过实训平台上的 USB 转串口模块实现串口收发功能。

（6）通过键盘、数码管和 LED 指示灯等完成人机交互操作。

（7）温度（*T*）、距离（*S*）测量结果刷新时间要求：

- 温度（*T*）≤0.5s。
- 距离（*S*）≤1s。

（8）距离（*S*）测量说明。

- 测量范围要求：10～50cm。
- 声音在空气中的传播速度：340m/s。

**2）显示功能**

（1）数据界面：包括温度显示、距离显示和变更次数显示等。

温度显示界面如图 4.55 所示，显示内容包括提示符 C 和温度值，温度值占用 4 位数码管，保留小数点后 2 位有效数字，单位为摄氏度（℃）。

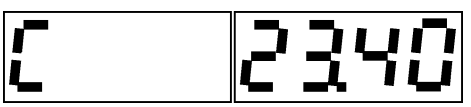



图 4.55　温度显示界面

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

距离显示界面如图 4.56 所示，显示内容包括提示符 L 和距离值，距离值占用 2 位数码管，距离值范围 0～99，单位为 cm。

图 4.56　距离显示界面

变更次数显示界面如图 4.57 所示，显示内容包括提示符 n 和参数变更次数，参数变更次数占用 5 位数码管，数据长度不足 5 位时，高位数码管熄灭，范围为 0～65535。

图 4.57　变更次数显示界面

（2）参数界面：包括温度参数和距离参数等。

温度参数界面如图 4.58 所示，显示内容包括提示符 P、参数编号 1 和温度参数，温度参数范围为 0～99。

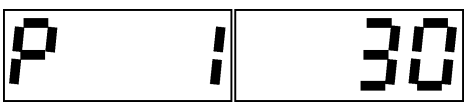



图 4.58　温度参数界面

距离参数界面如图 4.59 所示，显示内容包括提示符 P、参数编号 2 和距离参数，距离参数范围为 0～99。

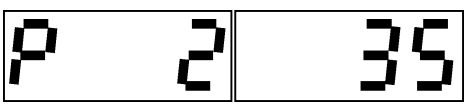



图 4.59　距离参数界面

**3）按键功能（为了保持一致，对按键功能进行了调整）**

（1）短按键功能说明如下。

S5：定义为界面按键，按下此按键，循环切换数据界面和参数界面，数据界面的默认显示为温度数据，参数界面的默认显示为温度参数。

S4：定义为切换按键，在数据界面按下此按键，循环切换显示温度数据、距离数据和参数变更次数；在参数界面下按下此按键，循环切换显示温度参数和距离参数。

S9：定义为加按键，在温度参数界面下按下此按键，温度参数增加 2℃；在距离参数界面按下此按键，距离参数增加 5cm。

S8：定义减按键，在温度参数界面按下此按键，温度参数减少 2℃；在距离参数界面按下此按键，距离参数减少 5cm。

（2）长按键功能说明如下。

任何界面下长按 S8 将参数变动次数重置为 0，长按 S9 启动或停止 DAC 输出。

（3）其他要求。

- 长按键触发时间要求：按键按下时间超过 1s，触发长按键功能，否则为短按键。
- 按键应消抖处理，避免出现一次按键功能多次触发等问题。
- 按键长按、短按对应的功能和效果不可互相影响。



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

● 请严格按照以上要求定义各按键长按和短按功能。

**4）存储功能**

（1）数据存储。

每当一次参数设置操作完成（从参数界面退出，切换回数据界面），如果参数发生变化，参数变动次数加 1，将参数变动次数保存在 EERPOM 存储器，可记录范围为 0～65535。

（2）数据重置。

参数变动次数可以通过长按 S12 重置为 0 次。

**5）串口功能**

（1）通信格式。

波特率：4800bps；数据位数：8 位；校验位：无校验；停止位：1 位。

（2）数据召测功能。

通过串口调试软件下发送数据召测指令，设备接收到正确指令后，上报数据或参数。召测指令格式要求如下。

● 查询数据指令：“ST\r\n”，接收到指令后，设备返回距离数据和温度数据，数据以字符串形式输出，以“$”开头，“\r\n”结尾，距离数据（*S*）和温度数据（*T*）以“,”分隔。

格式：$距离数据,温度数据\r\n

举例：$20,24.32\r\n

上报的距离数据为整数，温度数据保留小数点后两位有效数字。

● 查询参数指令：“PARA\r\n”，接收到指令后，设备返回当前的距离参数和温度参数，数据以字符串形式输出，以“#”开头，“\r\n”结尾，距离参数（S）和温度参数（T）以“,”分隔。

格式：#距离参数,温度参数\r\n

举例：#35,30\r\n

● 如设备接收到错误指令，返回“ERROR\r\n”。

**备注：**

（1）串口查询参数、数据指令响应时间要求：≤500ms。

（2）串口逻辑功能错乱，发送乱码和错误数据将被酌情扣分。

（3）S12、S13、S16 和 S17 按键扫描过程和串口通信功能应互不影响。

**6）DAC 输出功能**

（1）在启动状态下，DAC 输出电压值 $V_{OUT}$ 取决于测距数据（S），对应关系如下：

*S*≤距离参数时，$V_{OUT}$ = 2V。

*S*>距离参数时，$V_{OUT}$ = 4V。

（2）在停止状态下，DAC 固定输出 0.4V。

**7）LED 指示灯功能**

（1）温度指示灯。

当温度数据超过温度参数时，指示灯 L1 点亮，否则熄灭。

（2）距离指示灯。

当距离数据小于距离参数时，指示灯 L2 点亮，否则熄灭。

（3）DAC 功能指示灯。

启动状态下，指示灯 L3 点亮，停止状态下，指示灯 L3 熄灭。

（4）本试题未涉及的 LED 应处于熄灭状态，不同功能的指示灯状态切换时应互不影响。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

8）初始状态说明

请严格按照以下要求设计作品的上电初始状态。

（1）DAC 处于启动状态。

（2）作品上电后，未经任何操作的状态下，数码管处于数据界面下，显示温度数据。

（3）工作参数在每次上电时重置为默认值。

- 温度参数：30℃。
- 距离参数：35cm。

### 4.8.1 系统设计

系统详细框图如图 4.60 所示。单片机采集温度数据和距离数据、控制 DAC 输出、从键盘采集功能要求，并将温度和距离等数据按要求显示到数码管，控制 LED 灯点亮与熄灭，通过串口接收召测指令和发送数据及参数，24C02 作为 EEPROM 存储参数变更次数。

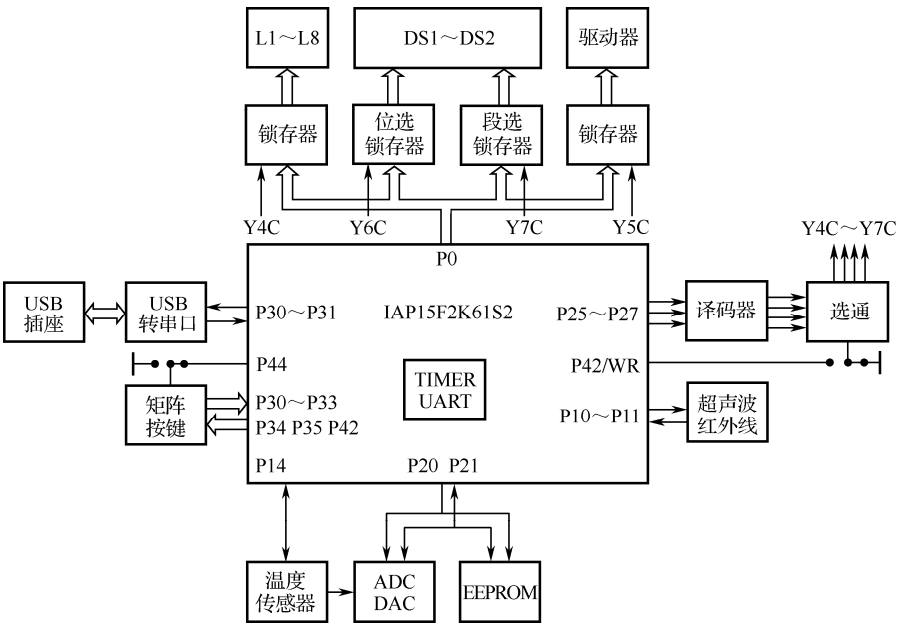



图 4.60　系统详细框图

系统主程序流程图如图 4.61 所示。主程序首先关闭外设、对定时器 1、定时器 0 和串口进行初始化并设置参数，然后进行按键处理、数码管处理和串口处理，其中按键处理中包含存储变更次数，数码管处理中包含温度、距离测量和 DAC 输出，串口处理包含接收字符处理和发送字符串，接收字符用中断方式实现。

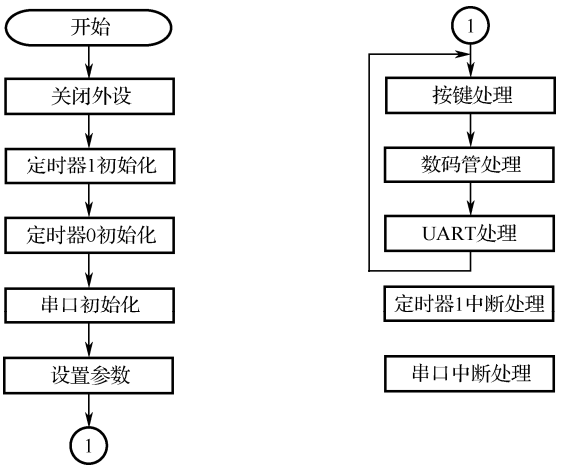



图 4.61　系统主程序流程图

系统“main.c”内容如下：

```
#include "tim.h"
#include "key.h"
#include "seg.h"
#include "stdio.h"
#include "onewire.h"
```



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
#include "ultrasonic.h"
#include "iic.h"
#include "uart.h"

unsigned int  uiTem_Val, uiSeg_Dly;
unsigned char ucDis_Val, ucNum;
unsigned char ucDis_Par, ucTem_Par;
unsigned char ucDis_Old, ucTem_Old;
unsigned char ucState, ucLed, ucDac=1;
unsigned char pucSeg_Buf[10], pucSeg_Code[8], ucSeg_Pos;
unsigned char ucKey_Dly, ucKey_Old, ucKey_Long;
unsigned char pucUart_Buf[12], ucUart_Num;
unsigned long ulms, ulKey_Time;

void Key_Proc(void);
void Seg_Proc(void);
void Uart_Proc(void);

void main(void)
{
  Cls_Peripheral();
  Timer1Init();
  Timer0Init();
  UartInit();

  EEPROM_Read(pucSeg_Buf, 0, 1);
  ucNum = pucSeg_Buf[0];
  ucTem_Par = 30;
  ucDis_Par = 35;

  while(1)
  {
    Key_Proc();
    Seg_Proc();
//  Uart_Proc();      // 为了便于在线调试，可以在其他功能调通后再加入 UART 处理
  }
}

void Time_1(void) interrupt 3
{
  ulms++;
  if(++ucKey_Dly == 10) ucKey_Dly = 0;
  if(++uiSeg_Dly == 500) uiSeg_Dly = 0;
  Led_Disp(ucLed);
  Seg_Disp(pucSeg_Code, ucSeg_Pos);
  if(++ucSeg_Pos == 8) ucSeg_Pos = 0;
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
}

void Key_Proc(void)
{
  unsigned char ucKey_Val, ucKey_Down, ucKey_Up;

  if(ucKey_Dly) return;
  ucKey_Dly = 1;

  ucKey_Val = Key_Read();
  ucKey_Down = ucKey_Val & (ucKey_Old ^ ucKey_Val);
  ucKey_Up = ~ucKey_Val & (ucKey_Old ^ ucKey_Val);
  ucKey_Old = ucKey_Val;

  if(ucKey_Down)
  {
    ulKey_Time = ulms;
    ucKey_Long = 0;
  }

  if(ulms-ulKey_Time < 1000)
    switch(ucKey_Up)
    {
      case 5:                     // S5
        if(ucState < 3)
        {
          ucState = 0x10;
          ucTem_Old = ucTem_Par;
          ucDis_Old = ucDis_Par;
        }
        else
        {
          ucState = 0;
          if((ucTem_Old != ucTem_Par) || (ucDis_Old != ucDis_Par))
          {
            pucSeg_Buf[0] = ++ucNum;
            EEPROM_Write(pucSeg_Buf, 0, 1);
          }
        }break;
      case 4:                     // S4
        if(ucState < 3)
        {
          if(++ucState == 3) ucState = 0;
        }
        else
          if(++ucState == 0x12) ucState = 0x10;
```



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
      break;
    case 9:                     // S9
      if(ucState == 0x10)
      {
        ucTem_Par += 2;
        if(ucTem_Par >= 100) ucTem_Par = 0;
      }
      if(ucState == 0x11)
      {
        ucDis_Par += 5;
        if(ucDis_Par >= 100) ucDis_Par = 0;
      }break;
    case 8:                     // S8
      if(ucState == 0x10)
      {
        if(ucTem_Par < 2) ucTem_Par = 100;
        ucTem_Par -= 2;
      }
      if(ucState == 0x11)
      {
        if(ucDis_Par < 5) ucDis_Par = 100;
        ucDis_Par -= 5;
      }
  }
  else
  {
    if((ucKey_Old == 9) && !ucKey_Long)
    {
      ucKey_Long = 1;
      ucDac ^= 1;
      if(ucDac) ucLed |= 4;
      else ucLed &= ~4;
    }
    if((ucKey_Old == 8) && ucNum)
    {
      ucNum = 0;
      pucSeg_Buf[0] = ucNum;
      EEPROM_Write(pucSeg_Buf, 0, 1);
    }
  }
}

void Seg_Proc(void)
{
  if(uiSeg_Dly) return;
  uiSeg_Dly = 1;
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
  switch(ucState)
  {
    case 0:                          // 温度数据
      uiTem_Val = rd_temperature();
      if(uiTem_Val > (ucTem_Par<<4)) ucLed |= 1;
      else ucLed &= ~1;
      sprintf(pucSeg_Buf, "C   %04.2f", (float)uiTem_Val/16.0);
      break;
    case 1:                          // 距离数据
      ucDis_Val = Wave_Recv();
      if(ucDis_Val < ucDis_Par) ucLed |= 2;
      else ucLed &= ~2;
      if(ucDac)
      {
        if(ucDis_Val <= ucDis_Par)
          PCF8591_Dac(102);          // 255*2/5
        else
          PCF8591_Dac(204);          // 255*4/5
      }
      else
         PCF8591_Dac(20);            // 255*0.4/5
      sprintf(pucSeg_Buf, "L    %3u", (unsigned int)ucDis_Val);
      break;
    case 2:                          // 变更次数
      sprintf(pucSeg_Buf, "N    %02u", (unsigned int)ucNum);
      break;
    case 0x10:                       // 温度参数
      sprintf(pucSeg_Buf, "P  1  %02u", (unsigned int)ucTem_Par);
      break;
    case 0x11:                       // 距离参数
      sprintf(pucSeg_Buf, "P  2  %02u", (unsigned int)ucDis_Par);
  }
  Seg_Tran(pucSeg_Buf, pucSeg_Code);
}

void Uart_Proc(void)
{
  if(ucUart_Num > 0)
  {
    if(pucUart_Buf[ucUart_Num-1] == 0xa)
    {
      if((pucUart_Buf[0]=='S') && (pucUart_Buf[1]=='T')\
        && (pucUart_Buf[2]==0xd))
        sprintf(pucUart_Buf, "$%02u,%04.2f\r\n",\
          (unsigned int)ucDis_Val, (float)uiTem_Val/16.0);
```



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
      else if((pucUart_Buf[0]=='P') && (pucUart_Buf[1]=='A')\
        && (pucUart_Buf[2]=='R') && (pucUart_Buf[3]=='A')\
        && (pucUart_Buf[4]==0xd))
        sprintf(pucUart_Buf, "#%02u,%02u\r\n",\
          (unsigned int)ucDis_Par, (unsigned int)ucTem_Par);
      else
        sprintf(pucUart_Buf, "ERROR\r\n");
      Uart_SendString(pucUart_Buf);
      ucUart_Num = 0;
    }
    else
      if(ucUart_Num == 6)
      {
        Uart_SendString("ERROR\r\n");
        ucUart_Num = 0;
      }
  }
}

void uart_0(void) interrupt 4
{
  if(RI)
  {
    pucUart_Buf[ucUart_Num++] = SBUF;
    RI = 0;
  }
}
```

### 4.8.2 系统实现

系统实现的主要步骤如下:

（1）在“D:\CT107D”文件夹中将“208_DS18B20”或“304_082”文件夹复制并重命名为“306_092”文件夹，进入“306_092”文件夹，双击工程文件“IAP15”打开工程。

（2）将“iic.c”“ultrasonic.c”和“uart.c”或“onewire.c”“uart.c”添加到“Source Group 1”中。

（3）修改“main.c”的内容。

（4）在“Options for Target”的“C51”标签中定义预处理符号“PCF8591_DAC EEPROM_AT24C02”。

（5）编译并下载程序。

**注意：**超声波测距时需要将 J2 的 1-3 和 2-4 短接。

### 4.8.3 客观题

（1）下列语句中，可将单片机 P2 口低三位状态取反的是（　　）。

A．P2 &= 0xF8　　B．P2 = ~P2

C．P2 ^= 0x07　　D．P2 |= 0x07

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

（2）RLC 串联电路的谐振频率为 1000Hz，当频率为 800Hz 的正弦电压源激励时该电路呈（　　）。

A．容性　　B．感性

C．阻性　　D．不能确定

（3）串口通信用波特率表示数据的传输速度，波特率表示的是（　　）。

A．帧/秒　　B．字符/秒

C．字节/秒　　D．位/秒

（4）三态门的输出状态包括（　　）。

A．高电平　　B．低电平

C．模拟输出　　D．高阻态

（5）有源二端口网络在端口开路时测得端口电压为 24V，当端口接入 10Ω 电阻时测得端口电压为 10V，则该二端网络的等效电动势为（　　），等效内阻为（　　）。

A．24V，10Ω　　B．24V，14Ω

C．10V，10Ω　　D．10V，14Ω

（6）MCS-51 单片机中，通常一些中间计算结果放在（　　）中。

A．累加器　　B．PC 寄存器

C．程序存储器　　D．DPTR 寄存器

（7）理想二极管电路如图 4.62 所示，A、B 两端的电压 $U_{AB}$ 为（　　）。

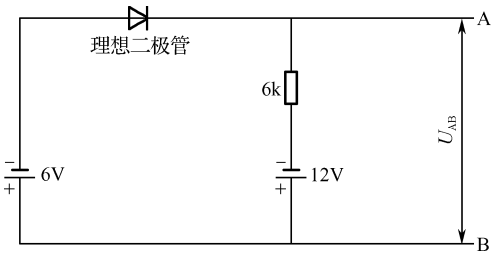



图 4.62　理想二极管电路

A．18V　　B．+12V

C．−6V　　D．−12V

（8）两个逻辑函数恒等，则它们必然具有唯一的（　　）。

A．逻辑表达式　　B．真值表

C．电路图　　D．逻辑图形符号

（9）全双工串行通信是指（　　）。

A．设计有数据发送和数据接收引脚　　B．发送与接收不互相制约

C．设计有两条数据传输线　　D．通信模式和速度可编程、可配置

（10）以下关于差分信号的说法中正确的是（　　）。

A．差分信号在 PCB 布线处理中，一般要求等间距、等长处理

B．CAN、RS-485、RS-232 均为差分接收方式，具有较好的抑制共模干扰能力

C．采用差分信号进行数据传输的通信方式中，数据接收端与发送端必须共地

D．数据接收端通过比较驱动端发送的两个电压信号差值来判断逻辑状态

（11）分析运算放大器的依据是（　　）。

A．$U_P \approx U_N$　　B．$A_U = 1$

C．$U_I = U_O$　　D．$I_N \approx I_P \approx 0$



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

（12）运算放大器电路如图 4.63 所示，$U_I$ = 2V，运放通过 DC 12V 单电源供电，输出端电压 $U_O$ 为（　　）。

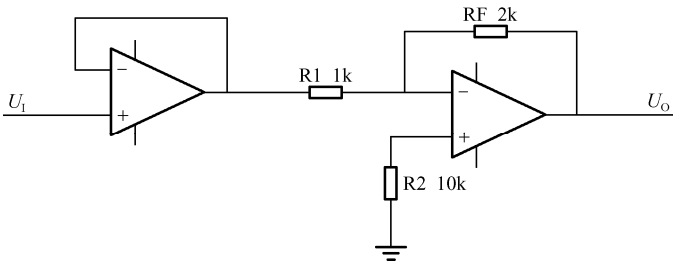



图 4.63　运算放大器电路

A．4V　　B．1V

C．−4V　　D．以上均不正确

（13）不具有压电效应的滤波器是（　　）。

A．石英晶体滤波器　　B．LC 滤波器

C．RC 滤波器　　D．声表面波滤波器

（14）以下关于 IAP15F2K61S2 单片机的说法中错误的是（　　）。

A．所有 IO 口都具有 4 种工作模式

B．IO 口最大翻转速度为系统时钟

C．低优先级中断可以被高优先级中断所中断，可现实 2 级中断服务程序嵌套

D．通过外部中断检测下降沿，要求信号在相应引脚上维持高低电平超过 1 个时钟周期

（15）下列有关信号的说法中错误的是（　　）。

A．信号是消息的表现形式　　B．信号都可以用一个确定的时间函数来描述

C．声音和图像都是信号　　D．信号可以分解为周期信号和非周期信号

参考答案与评分标准

（1）BC（2 分）　（2）A（2 分）　（3）D（2 分）　（4）ABD（2 分）　（5）B（2 分）

（6）A（2 分）　（7）C（2 分）　（8）B（2 分）　（9）B（2 分）　（10）AD（2 分）

（11）AD（2 分）　（12）D（2 分）　（13）BC（2 分）　（14）B（2 分）　（15）B（2 分）

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

# 第 5 章　嵌入式设计与开发项目模块程序设计

本章介绍嵌入式设计与开发项目模块程序设计，包括头文件、源文件和范例程序。为了使用方便，头文件存放在“D:\CT117E\inc”文件夹中，源文件存放在“D:\CT117E\sre”文件夹中，范例工程在模板工程的基础上实现，存放在“D:\CT117E”文件夹中。

嵌入式设计与开发项目模块程序设计包括下列内容：

（1）LED 指示灯程序设计。

（2）独立按键扫描程序设计。

（3）LCD 应用程序设计。

（4）USART 应用程序设计。

（5）AT24C02 存储器应用程序设计。

（6）ADC 应用程序设计。

（7）TIM 应用程序设计。

（8）数码管静态显示程序设计。

（9）ADC 键盘扫描程序设计。

（10）DHT11 湿度传感器程序设计。

（11）DS18B20 温度传感器程序设计。

（12）LIS302DL 加速度传感器程序设计。

## 5.1　LED 指示灯程序设计

实训平台上的 8 个 LED 指示灯 LD1～LD8 通过锁存器分别与 PC8～PC15 相连，低电平时 LED 亮，高电平时 LED 灭。锁存器的使能端与 PD2 相连，低电平时锁存数据（锁存器输出不随输入变化），高电平时不锁存数据（锁存器输出随输入变化）。

LED 头文件“led.h”内容如下：

```
#include "stm32f10x.h"

void LED_Init(void);
void LED_Disp(unsigned char ucLed);
void BUZ_Init(void);
```

LED 源文件“led.c”内容如下：

```
#include "led.h"
// LED 接口初始化
void LED_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct;
  // 允许 GPIOC 和 GPIOD 时钟
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC, ENABLE);
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOD, ENABLE);
```



电子工业出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
  // PC8~PC15 通用推挽输出
  GPIO_InitStruct.GPIO_Pin = GPIO_Pin_8 | GPIO_Pin_9 |\
    GPIO_Pin_10 | GPIO_Pin_11 | GPIO_Pin_12 | GPIO_Pin_13 |\
    GPIO_Pin_14 | GPIO_Pin_15;
  GPIO_InitStruct.GPIO_Speed = GPIO_Speed_2MHz;
  GPIO_InitStruct.GPIO_Mode = GPIO_Mode_Out_PP;
  GPIO_Init(GPIOC, &GPIO_InitStruct);
  // PD2 通用推挽输出
  GPIO_InitStruct.GPIO_Pin = GPIO_Pin_2;
  GPIO_Init(GPIOD, &GPIO_InitStruct);
}
// LED 显示
// 入口参数：ucLed-8 位显示控制，1-亮，0-灭
void LED_Disp(unsigned char ucLed)
{
  GPIO_Write(GPIOC, ~ucLed << 8);
  GPIO_SetBits(GPIOD,GPIO_Pin_2);
  GPIO_ResetBits(GPIOD,GPIO_Pin_2);
}
// 蜂鸣器接口初始化
void BUZ_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct;
  // 允许 AFIO 和 GPIOB 时钟
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO, ENABLE);
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);
  // 配置引脚映射（禁止 JTRST 功能）
  GPIO_PinRemapConfig(GPIO_Remap_SWJ_NoJTRST, ENABLE);
  // PB4 推挽输出
  GPIO_InitStruct.GPIO_Pin = GPIO_Pin_4;
  GPIO_InitStruct.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_InitStruct.GPIO_Mode = GPIO_Mode_Out_PP;
  GPIO_Init(GPIOB, &GPIO_InitStruct);
}
```

竞赛实训平台上的蜂鸣器与 PB4 相连，低电平时打开蜂鸣器，高电平时关闭蜂鸣器。

**注意**：由于 PB4 的复位功能是 JTRST（JTAG 复位），所以初始化时必须先用配置引脚映射函数 GPIO_PinRemapConfig()禁止 JTRST 功能，而引脚映射通过 AFIO 实现，因此还必须允许 AFIO 时钟。

LED 范例程序主文件“main.c”内容如下：

```
#include "led.h"

unsigned char ucLed = 1;
unsigned long ulTick_ms;

int main(void)
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
{
  SysTick_Config(72000);          // 定时 1ms(HCLK = 72MHz)
  LED_Init();

  while(1);
}

void SysTick_Handler(void)
{
  ulTick_ms++;

  if(ulTick_ms == 1000)
  {
    ulTick_ms = 0;

    ucLed <<= 1;
    if(ucLed == 0) ucLed = 1;
    LED_Disp(ucLed);
  }
}
```

LED 范例程序的实现步骤如下：

（1）将模板工程文件夹“Project”复制并重命名为“201_LED”文件夹，在“201_LED”文件夹中双击工程文件“STM32.uvproj”打开工程。

（2）在“D:\CT117E\inc”文件夹中新建文件“led.h”。

（3）在“D:\CT117E\sre”文件夹中新建文件“led.c”。

（4）在“201_LED”文件夹新建文件“main.c”。

（5）将下列库文件添加到“Library”组：

- stm32f10x_rcc.c
- stm32f10x_gpio.c

（6）将“led.c”添加到“Driver”组。

（7）将“main.c”添加到“User”组。

## 5.2 独立按键扫描程序设计

实训平台上的 4 个独立按键 B1~B4 分别与 PA0、PA8、PB1 和 PB2 相连，按键未按下时为高电平，按键按下时为低电平。

独立按键头文件“key.h”内容如下：

```
#include "stm32f10x.h"

void KEY_Init(void);
unsigned char KEY_Scan(void);
void Delay_KEY(unsigned int ms);
```

独立按键源文件“key.c”内容如下：



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
#include "key.h"
// KEY 接口初始化
void KEY_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct;
  // 允许 GPIOA 和 GPIOB 时钟
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);
  // PA0 和 PA8 浮空输入（复位状态，可以省略）
  GPIO_InitStruct.GPIO_Pin = GPIO_Pin_0 | GPIO_Pin_8;
  GPIO_InitStruct.GPIO_Mode = GPIO_Mode_IN_FLOATING;
  GPIO_Init(GPIOA, &GPIO_InitStruct);
  // PB1 和 PB2 浮空输入（复位状态，可以省略）
  GPIO_InitStruct.GPIO_Pin = GPIO_Pin_1 | GPIO_Pin_2;
  GPIO_InitStruct.GPIO_Mode = GPIO_Mode_IN_FLOATING;
  GPIO_Init(GPIOB, &GPIO_InitStruct);
}
// KEY 扫描
unsigned char KEY_Scan(void)
{
  unsigned char ucKey_Val = 0;
  // 判断 B1 和 B2 是否按下
  if(~GPIO_ReadInputData(GPIOA) & 0x101)
  {
    Delay_KEY(10);                                          // 延时 10ms 消抖
    if(!GPIO_ReadInputDataBit(GPIOA, GPIO_Pin_0))           // B1 按下
      ucKey_Val = 1;
    if(!GPIO_ReadInputDataBit(GPIOA, GPIO_Pin_8))           // B2 按下
      ucKey_Val = 2;
  }
  // 判断 B3 和 B4 是否按下
  else if(~GPIO_ReadInputData(GPIOB) & 6)
  {
    Delay_KEY(10);                                          // 延时 10ms 消抖
    if(!GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_1))           // B3 按下
      ucKey_Val = 3;
    if(!GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_2))           // B4 按下
      ucKey_Val = 4;
  }
  return ucKey_Val;
}

void Delay_KEY(unsigned int ms)
{
  unsigned int i, j;
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
  for(i=0; i<ms; i++)
    for(j=0; j<7992; j++);            // SYSCLK = 72MHz
//  for(j=0; j<1598; j++);            // SYSCLK = 8MHz
}
```

独立按键范例程序主文件“main.c”内容如下：

```
#include "key.h"
#include "led.h"

unsigned char ucLed = 1;
unsigned char ucKey_Long;
unsigned long ulTick_ms, ulKey_Time;

void KEY_Proc(void);

int main(void)
{
  SysTick_Config(72000);          // 定时 1ms(HCLK = 72MHz)
  KEY_Init();
  LED_Init();
  BUZ_Init();

  while(1)
  {
    KEY_Proc();
    LED_Disp(ucLed);
  }
}

void KEY_Proc(void)
{
  unsigned char ucKey_Val;

  ucKey_Val = KEY_Scan();
  if(ucKey_Val != ucKey_Long)
  {
    ucKey_Long = ucKey_Val;
    ulKey_Time = ulTick_ms;
  }
  else
    ucKey_Val = 0;

  if(ucKey_Val == 1)              // B1 短按
  {
    ucLed <<= 1;
    if(ucLed == 0) ucLed = 1;
```



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
  }
  if(ucKey_Val == 2)              // B2 短按
  {
    ucLed >>= 1;
    if(ucLed == 0) ucLed = 0x80;
  }
  if(ucKey_Long == 1)             // B1 长按
  {
    if(ulTick_ms-ulKey_Time > 800)
    {
      ulKey_Time = ulTick_ms;
      ucLed <<= 2;
      if(ucLed == 0) ucLed = 1;
    }
  }
  if(ucKey_Long == 2)             // B2 长按
  {
    if(ulTick_ms-ulKey_Time > 800)
    {
//    ulKey_Time = ulTick_ms;
      ucLed >>= 2;
      if(ucLed == 0) ucLed = 0x80;
    }
  }
  if(ucKey_Long == 3)             // B3 长按
    GPIO_ResetBits(GPIOB, GPIO_Pin_4);
  else
    GPIO_SetBits(GPIOB, GPIO_Pin_4);
}
// SysTick 中断处理
void SysTick_Handler(void)
{
  ulTick_ms++;
}
```

独立按键范例程序的实现在 LED 范例程序实现的基础上修改完成，步骤如下：

（1）将“201_LED”文件夹复制并重命名为“202_KEY”文件夹，在“202_KEY”文件夹中双击工程文件“STM32.uvproj”打开工程。

（2）在“inc”文件夹中新建文件“key.h”。

（3）在“sre”文件夹中新建文件“key.c”。

（4）将“key.c”添加到“Driver”组。

（5）修改文件“main.c”。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

## 5.3 LCD 应用程序设计

LCD 是低功耗显示器件，可以通过并口控制，也可以通过串口控制。嵌入式竞赛实训平台使用的是 240×320 TFT LCD，通过并行接口与 STM32 相连，连接关系如表 5.1 所示。

表 5.1 LCD 与 STM32 连接关系

| LCD 引脚 | STM32 引脚 | STM32 方向 | 说　明 |
|---|---|---|---|
| CS# | PB9 | 输出 | 片选（低电平有效） |
| RS | PB8 | 输出 | 寄存器选择：0—索引或状态寄存器，1—控制寄存器 |
| WR# | PB5 | 输出 | 写选通（低电平有效） |
| RD# | PB10 | 输出 | 读选通（低电平有效，与 SD 卡的 SD1 公用） |
| PD1~PD8 | PC0~PC7 | 双向 | 数据低 8 位 |
| PD10~PD17 | PC8~PC15 | 双向 | 数据高 8 位 |

LCD 库函数在“lcd.h”中声明，分为低层库函数、中层库函数和高层库函数。

低层库函数用于配置连接 LCD 的 GPIO 引脚，在“lcd.h”中声明如下：

```
void LCD_CtrlLinesConfig(void);
void LCD_BusIn(void);
void LCD_BusOut(void);
```

LCD_CtrlLinesConfig()首先将 PB5（LCD_WR#）、PB8（LCD_RS）、PB9（LCD_CS#）和 PB10（LCD_RD#）配置为通用推挽输出（GPIO_Mode_Out_PP），然后调用 LCD_BusOut()将 PC0~PC15（LCD_PD）配置为通用推挽输出，最后将 PB5（LCD_WR#）、PB9（LCD_CS#）和 PB10（LCD_RD#）设置为高电平。

LCD_BusIn()和 LCD_BusOut()分别将 PC0~PC15（LCD_PD）配置为上拉输入（GPIO_Mode_IPU）和通用推挽输出（GPIO_Mode_Out_PP）。

中层库函数主要用于 LCD 控制器中寄存器和显示缓存的读写，在“lcd.h”中声明如下：

```
void LCD_WriteReg(u8 LCD_Reg, u16 LCD_RegValue);
u16 LCD_ReadReg(u8 LCD_Reg);
void LCD_WriteRAM_Prepare(void);
void LCD_WriteRAM(u16 RGB_Code);
u16 LCD_ReadRAM(void);
void LCD_PowerOn(void);
void LCD_DisplayOn(void);
void LCD_DisplayOff(void);
```

高层库函数供用户调用，常用高层库函数在“lcd.h”中声明如下：

```
void STM3210B_LCD_Init(void);
void LCD_SetTextColor(vu16 Color);
void LCD_SetBackColor(vu16 Color);
void LCD_ClearLine(u8 Line);
void LCD_Clear(u16 Color);
void LCD_SetCursor(u8 Xpos, u16 Ypos);
```



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
void LCD_DrawChar(u8 Xpos, u16 Ypos, uc16 *c);
void LCD_DisplayChar(u8 Line, u16 Column, u8 Ascii);
void LCD_DisplayStringLine(u8 Line, u8 *ptr);
void LCD_DrawLine(u8 Xpos, u16 Ypos, u16 Length, u8 Direction);
void LCD_DrawRect(u8 Xpos, u16 Ypos, u8 Height, u16 Width);
void LCD_DrawCircle(u8 Xpos, u16 Ypos, u16 Radius);
```

LCD 范例程序主文件“main.c”内容如下：

```
#include "key.h"
#include "led.h"
#include "lcd.h"
#include "stdio.h"

unsigned char ucSec;
unsigned char pucStr[21];
unsigned long ulTick_ms;

void LCD_Proc(void);

int main(void)
{
  SysTick_Config(72000);          // 定时 1ms(HCLK = 72MHz)
  KEY_Init();
  LED_Init();

  STM3210B_LCD_Init();
  LCD_Clear(Blue);
  LCD_SetBackColor(Blue);
  LCD_SetTextColor(White);

  while(1)
  {
    LED_Disp(ucSec);
    LCD_Proc();
  }
}

void LCD_Proc(void)
{
  LCD_DisplayChar(Line3, 176, ucSec/10+0x30);
  LCD_SetBackColor(Red);
  LCD_DisplayChar(Line3, 160, ucSec%10+0x30);
  LCD_SetBackColor(Blue);

  pucStr[9] = ucSec/10+0x30;
  pucStr[10] = ucSec%10+0x30;
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
  LCD_DisplayStringLine(Line5, pucStr);

  sprintf((char *)pucStr, "       %02u       ", ucSec);
  LCD_SetBackColor(Red);
  LCD_DisplayStringLine(Line7, pucStr);
  LCD_SetBackColor(Blue);
}
// SysTick 中断处理程序
void SysTick_Handler(void)
{
  ulTick_ms++;
  if(ulTick_ms%1000 == 0) ucSec++;
}
```

LCD 范例程序的实现在 KEY 范例程序实现的基础上修改完成，步骤如下：

（1）将竞赛资源包“液晶驱动参考例程”中的下列文件复制到“D:\CT117E\Inc”文件夹：

- lcd.h：LCD 头文件。
- fonts.h：ASCII 字符点阵数据文件。

（2）将竞赛资源包“液晶驱动参考例程”中的下列文件复制到“D:\CT117E\Sre”文件夹：

- lcd.c：LCD 库文件。

（3）将“202_KEY”文件夹复制并重命名为“203_LCD”文件夹，在“203_LCD”文件夹中双击工程文件“STM32.uvproj”打开工程。

（4）将“lcd.c”添加到“Driver”组。

（5）修改文件“main.c”。

## 5.4 USART 应用程序设计

实训平台上的 UART1 通过电平转换与 9 针 D 型插座 CN6 相连，UART2 通过 UART-USB 转换与 USB 插座 CN2 相连。

USART 头文件“usart.h”内容如下：

```
#include "stm32f10x.h"
#include "stdio.h"

void USART2_Init(unsigned long ulBaud);
unsigned char USART_SendChar(USART_TypeDef* USARTx, unsigned char ucChar);
void USART_SendString(USART_TypeDef* USARTx, unsigned char* pucStr);
```

USART 源文件“usart.c”内容如下：

```
#include "usart.h"

void USART2_Init(unsigned long ulBaud)
{
  GPIO_InitTypeDef GPIO_InitStruct;
  USART_InitTypeDef USART_InitStruct;
```



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
  NVIC_InitTypeDef NVIC_InitStruct;
  // 允许 GPIOA 和 USART2 时钟
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
  RCC_APB1PeriphClockCmd(RCC_APB1Periph_USART2, ENABLE);
  // PA2-TX2 复用推挽输出
  GPIO_InitStruct.GPIO_Pin = GPIO_Pin_2;
  GPIO_InitStruct.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_InitStruct.GPIO_Mode = GPIO_Mode_AF_PP;
  GPIO_Init(GPIOA, &GPIO_InitStruct);
  // PA3-TX2 浮空输入（复位状态，可以省略）
  GPIO_InitStruct.GPIO_Pin = GPIO_Pin_3;
  GPIO_InitStruct.GPIO_Mode = GPIO_Mode_IN_FLOATING;
  GPIO_Init(GPIOA, &GPIO_InitStruct);
  // 初始化 USART2（波特率 ulBaud，默认 8 个数据位，1 个停止位，无校验，允许 Rx 和 Tx）
  USART_InitStruct.USART_BaudRate = ulBaud;
  USART_InitStruct.USART_WordLength = USART_WordLength_8b;
  USART_InitStruct.USART_StopBits = USART_StopBits_1;
  USART_InitStruct.USART_Parity = USART_Parity_No;
  USART_InitStruct.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;
  USART_InitStruct.USART_HardwareFlowControl
    = USART_HardwareFlowControl_None;
  USART_Init(USART2, &USART_InitStruct);
  // 允许 USART2
  USART_Cmd(USART2, ENABLE);
  // 允许 USART2 接收中断
  USART_ITConfig(USART2, USART_IT_RXNE, ENABLE);
  // 允许 NVIC USART2 中断
  NVIC_InitStruct.NVIC_IRQChannel = USART2_IRQn;
  NVIC_InitStruct.NVIC_IRQChannelPreemptionPriority = 0;
  NVIC_InitStruct.NVIC_IRQChannelSubPriority = 0;
  NVIC_InitStruct.NVIC_IRQChannelCmd = ENABLE;
  NVIC_Init(&NVIC_InitStruct);
}
// 发送字符
unsigned char USART_SendChar(USART_TypeDef* USARTx, unsigned char ucChar)
{
  while(!USART_GetFlagStatus(USARTx, USART_FLAG_TXE));
  USART_SendData(USARTx, ucChar);
  return ucChar;
}
// 发送字符串
void USART_SendString(USART_TypeDef* USARTx, unsigned char* pucStr)
{
  while(*pucStr != '\0')
    USART_SendChar(USARTx, *pucStr++);
}
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
// printf 调用函数（使用 MicroLIB）
int fputc(int ch, FILE *f)
{
  return(USART_SendChar(USART2, ch));
}
```

**注意：**使用 printf()时需要使用 MicroLIB（在“目标选项”对话框“Target”（目标）标签的“Code Generation”（代码生成）下选择“Use MicroLIB”），否则程序不能正常工作。

USART 范例程序主文件“main.c”内容如下：

```
#include "key.h"
#include "led.h"
#include "lcd.h"
#include "usart.h"

unsigned char ucSec, ucSec1;
unsigned char pucStr[21];
unsigned char pucRcv[5] = "  \0  ", ucRno = 3;
unsigned long ulTick_ms;

void LCD_Proc(void);
void USART_Proc(void);

int main(void)
{
  SysTick_Config(72000);          // 定时 1ms(HCLK = 72MHz)
  KEY_Init();
  LED_Init();

  STM3210B_LCD_Init();
  LCD_Clear(Blue);
  LCD_SetBackColor(Blue);
  LCD_SetTextColor(White);
  USART2_Init(9600);

  while(1)
  {
    LED_Disp(ucSec);
    LCD_Proc();
    USART_Proc();
  }
}

void LCD_Proc(void)
{
  sprintf((char *)pucStr, "        %02u        ", ucSec);
  LCD_DisplayStringLine(Line5, pucStr);
```



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
}

void USART_Proc(void)
{
  if(ucSec != ucSec1)
  {
    ucSec1 = ucSec;

    USART_SendChar(USART2, ucSec/10+0x30);
    USART_SendChar(USART2, ucSec%10+0x30);
    USART_SendChar(USART2, ' ');

    pucRcv[0] = ucSec/10+0x30;
    pucRcv[1] = ucSec%10+0x30;
    USART_SendString(USART2, pucRcv);

    printf(" %s\r\n", pucRcv);
  }

  if(ucRno == 5)
  {
    ucRno = 3;
    ucSec = (pucRcv[3]-'0')*10 + pucRcv[4]-'0';
  }
}
// SysTick 中断处理程序
void SysTick_Handler(void)
{
  ulTick_ms++;
  if(ulTick_ms%1000 == 0) ucSec++;
}
// USART2 中断处理程序
void USART2_IRQHandler(void)
{
  pucRcv[ucRno++] = USART_ReceiveData(USART2);
}
```

USART 范例程序的实现在 LCD 范例程序实现的基础上修改完成，步骤如下：

（1）将“203_LCD”文件夹复制并重命名为“204_USART”文件夹，在“204_USART”文件夹中双击工程文件“STM32.uvproj”打开工程。

（2）在“inc”文件夹中新建文件“usart.h”。

（3）在“sre”文件夹中新建文件“usart.c”。

（4）将下列库文件添加到“Library”组：

- stm32f10x_usart.c
- misc.c

（5）将“usart.c”添加到“Driver”组。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

（6）修改文件“main.c”。

（7）在“目标选项”对话框“Target”（目标）标签的“Code Generation”（代码生成）下选择“Use MicroLIB”

## 5.5 AT24C02 存储器应用程序设计

AT24C02 是 2kbit 串行 EEPROM，内部组织为 256×8bit，支持 16Byte 页写，写周期内部定时（小于 5ms），2 线串行接口，可实现 8 个器件共用 1 个接口，工作电压为 2.7～5.5V，8 引脚封装，引脚说明如表 5.2 所示。

表 5.2　AT24C02 引脚说明

| 引　脚 | 功　能 | 方　向 | 说　明 | 引　脚 | 功　能 | 方　向 | 说　明 |
|---|---|---|---|---|---|---|---|
| 1 | A0 | 输入 | 器件地址 0 | 5 | SDA | 双向 | 串行数据 |
| 2 | A1 | 输入 | 器件地址 1 | 6 | SCL | 输入 | 串行时钟 |
| 3 | A2 | 输入 | 器件地址 2 | 7 | WP | 输入 | 写保护 |
| 4 | GND | - | 地 | 8 | VCC | 输入 | 电源（2.7～5.5V） |

AT24C02 的字节数据读写格式如图 5.1 所示。

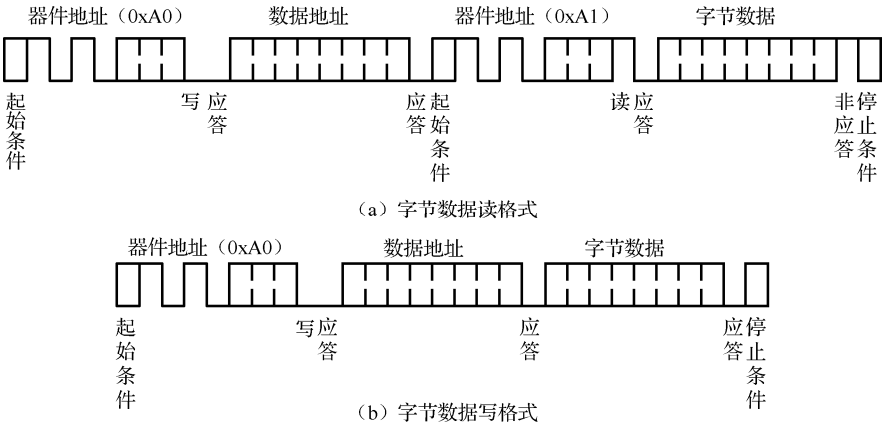



图 5.1　AT24C02 字节数据读写格式

AT24C02 的读写操作可在竞赛资源包“参考程序”文件夹中“i2c.h”和“i2c.c”的基础上修改实现，步骤如下：

（1）将竞赛资源包“参考程序”中的下列文件复制粘贴到“D:\CT117E\Inc”文件夹：

● i2c.h：I2C 头文件。

（2）将竞赛资源包“参考程序”中的下列文件复制粘贴到“D:\CT117E\Sre”文件夹：

● i2c.c：I2C 库文件。

（3）在“i2c.h”中追加下列声明：

```
void i2c_read(unsigned char* pucBuf, unsigned char ucAddr,
  unsigned char ucNum);
void i2c_write(unsigned char* pucBuf, unsigned char ucAddr,
  unsigned char ucNum);
```

（4）将“i2c.c”中 I2CWaitAck()函数后部的语句：



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
SDA_Output_Mode();
SCL_Output(0);delay1(500);
```

修改为（交换顺序，原顺序在高主频时会错误产生停止条件）：

```
SCL_Output(0);delay1(500);
SDA_Output_Mode();
```

（5）在“i2c.c”中追加下列内容：

```
// AT24C02 读
void i2c_read(unsigned char* pucBuf, unsigned char ucAddr,
  unsigned char ucNum)
{
  I2CStart();
  I2CSendByte(0xa0);
  I2CWaitAck();

  I2CSendByte(ucAddr);
  I2CWaitAck();

  I2CStart();
  I2CSendByte(0xa1);
  I2CWaitAck();

  while(ucNum--)
  {
    *pucBuf++ = I2CReceiveByte();
    if(ucNum)
      I2CSendAck();
    else
      I2CSendNotAck();
  }
  I2CStop();
}
// AT24C02 写
void i2c_write(unsigned char* pucBuf, unsigned char ucAddr,
  unsigned char ucNum)
{
  I2CStart();
  I2CSendByte(0xa0);
  I2CWaitAck();

  I2CSendByte(ucAddr);
  I2CWaitAck();

  while(ucNum--)
  {
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
    I2CSendByte(*pucBuf++);
    I2CWaitAck();
  }
  I2CStop();
  delay1(500);
}
```

AT24C02 读写操作范例程序主文件“main.c”内容如下：

```
#include "key.h"
#include "led.h"
#include "lcd.h"
#include "usart.h"
#include "i2c.h"

unsigned char ucSec;
unsigned char pucBuf1[16], pucBuf2[256];
unsigned long ulTick_ms;

int main(void)
{
  unsigned char i;

  SysTick_Config(72000);          // 定时 1ms(HCLK = 72MHz)
  KEY_Init();
  LED_Init();

  STM3210B_LCD_Init();
  LCD_Clear(Blue);
  LCD_SetBackColor(Blue);
  LCD_SetTextColor(White);

  USART2_Init(9600);
  i2c_init();

  for(i=0; i<14; i++)
    pucBuf1[i] = i+0x30;
  pucBuf1[14] = 0x0d;
  pucBuf1[15] = 0x0a;

  for(i=0; i<16; i++)
  {
    i2c_write(pucBuf1, i*16, 16);     // 页写（最多 16Byte）

    LED_Disp(i);
    LCD_DisplayChar(Line5, 176, i/10+0x30);
    LCD_DisplayChar(Line5, 160, i%10+0x30);
```



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
    printf("%02u ", i);
  }

  i2c_read(pucBuf2, 0, 255);
  printf("\r\n%s\r\n", pucBuf2);

  while(1);
}
// SysTick 中断处理程序
void SysTick_Handler(void)
{
  ulTick_ms++;
  if(ulTick_ms%1000 == 0) ucSec++;
}
```

AT24C02 读写范例程序的实现在 USART 范例程序实现的基础上修改完成，步骤如下：

（1）将“204_USART”文件夹复制并重命名为“205_AT24C02”文件夹，在“205_AT24C02”文件夹中双击工程文件“STM32.uvproj”打开工程。

（2）将“i2c.c”添加到“driver”组。

（3）修改文件“main.c”。

## 5.6 ADC 应用程序设计

ADC 有多达 18 个转换通道，其中通道 0~通道 15 是外部通道，使用的 GPIO 引脚如表 5.3 所示。

表 5.3 ADC 外部通道使用的 GPIO 引脚

| ADC 引脚 | GPIO 引脚 | GPIO 配置 | ADC 引脚 | GPIO 引脚 | GPIO 配置 |
|---|---|---|---|---|---|
| IN0 | PA0 | 模拟输入 | IN8 | PB0 | 模拟输入 |
| IN1 | PA1 | 模拟输入 | IN9 | PB1 | 模拟输入 |
| IN2 | PA2 | 模拟输入 | IN10 | PC0 | 模拟输入 |
| IN3 | PA3 | 模拟输入 | IN11 | PC1 | 模拟输入 |
| IN4 | PA4 | 模拟输入 | IN12 | PC2 | 模拟输入 |
| IN5 | PA5 | 模拟输入 | IN13 | PC3 | 模拟输入 |
| IN6 | PA6 | 模拟输入 | IN14 | PC4 | 模拟输入 |
| IN7 | PA7 | 模拟输入 | IN15 | PC5 | 模拟输入 |

ADC1 的通道 16 内部与温度传感器（推荐采样时间 17.1μs）相连，通道 17 内部与参考电源 $V_{REFINT}$ 相连，使用 ADC_TempSensorVrefintCmd()使能。

转换通道分为规则通道和注入通道两组。

规则通道由最多 16 个通道组成，按顺序转换，使用 ADC_RegularChannelConfig()进行配置，使用 ADC_GetConversionValue()获取转换值。

注入通道由最多 4 个通道组成，可插入转换，使用 ADC_InjectedChannelConfig()进行配置，使用 ADC_GetInjectedConversionValue()获取转换值。

ADC 通道配置的一个主要参数是采样时间，采样时间通过采样周期数进行配置，两者的关系是：

电子工业出版社 PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

采样时间= (采样周期数 + 12.5) / $f_{ADCCLK}$

采样周期数的取值有：1.5、7.5、13.5、28.5、41.5、55.5、71.5 和 239.5 等。

当 $f_{ADCCLK}$ 取最大值 14MHz 时，内部温度传感器的采样周期数是：

采样周期数 = 17.1μs ×14MHz − 12.5 = 226.9

实训平台上的通道 8 通过 PB0 与电位器 R37 相连。

ADC 头文件“adc.h”内容如下：

```
#include "stm32f10x.h"

void ADC1_Init(void);
unsigned int ADC1_Conv(void);
unsigned int ADC1_InjeConv(void);
```

ADC 源文件“adc.c”内容如下：

```
#include "adc.h"

void ADC1_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct;
  ADC_InitTypeDef ADC_InitStruct;
  // 允许 GPIOB 和 ADC1 时钟
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1, ENABLE);
  // PB0-IN8 模拟输入
  GPIO_InitStruct.GPIO_Pin = GPIO_Pin_0;
  GPIO_InitStruct.GPIO_Mode = GPIO_Mode_AIN;
  GPIO_Init(GPIOB, &GPIO_InitStruct);
  // 初始化 ADC1
  ADC_InitStruct.ADC_Mode = ADC_Mode_Independent;
  ADC_InitStruct.ADC_ScanConvMode = DISABLE;
  ADC_InitStruct.ADC_ContinuousConvMode = DISABLE;
  ADC_InitStruct.ADC_ExternalTrigConv = ADC_ExternalTrigConv_None;
  ADC_InitStruct.ADC_DataAlign = ADC_DataAlign_Right;
  ADC_InitStruct.ADC_NbrOfChannel = 1;
  ADC_Init(ADC1, &ADC_InitStruct);
  // 配置通道 8
  ADC_RegularChannelConfig(ADC1, ADC_Channel_8, 1,
    ADC_SampleTime_1Cycles5);
  // 配置通道 16
  ADC_InjectedChannelConfig(ADC1, ADC_Channel_16, 1,
    ADC_SampleTime_239Cycles5);
  ADC_TempSensorVrefintCmd(ENABLE);
  ADC_AutoInjectedConvCmd(ADC1, ENABLE);
  // 启动 ADC1
  ADC_Cmd(ADC1, ENABLE);
  // 校准 ADC1
```



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
  ADC_StartCalibration(ADC1);
  while(ADC_GetCalibrationStatus(ADC1));
}

unsigned int ADC1_Conv(void)
{
  ADC_SoftwareStartConvCmd(ADC1, ENABLE);
  while(!ADC_GetFlagStatus(ADC1, ADC_FLAG_EOC));
  return ADC_GetConversionValue(ADC1);
}

unsigned int ADC1_InjeConv(void)
{
  ADC_SoftwareStartConvCmd(ADC1, ENABLE);
  while(!ADC_GetFlagStatus(ADC1, ADC_FLAG_JEOC));
  ADC_ClearFlag(ADC1, ADC_FLAG_JEOC);
  return ADC_GetInjectedConversionValue(ADC1, ADC_InjectedChannel_1);
}
```

ADC 范例程序主文件“main.c”内容如下：

```
#include "key.h"
#include "led.h"
#include "lcd.h"
#include "usart.h"
#include "i2c.h"
#include "adc.h"

unsigned int  uiAdc_Val;
unsigned char ucSec, ucSec1;
unsigned char pucStr[21];
unsigned long ulTick_ms;

void ADC_Proc(void);

int main(void)
{
  SysTick_Config(72000);          // 定时 1ms(HCLK = 72MHz)
  KEY_Init();
  LED_Init();

  STM3210B_LCD_Init();
  LCD_Clear(Blue);
  LCD_SetBackColor(Blue);
  LCD_SetTextColor(White);

  USART2_Init(9600);
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
  i2c_init();
  ADC1_Init();

  sprintf((char *)pucStr, "     ADC DEMO");
  LCD_DisplayStringLine(Line3, pucStr);
  printf("%s\r\n", pucStr);

  while(1)
  {
    LED_Disp(ucSec);
    ADC_Proc();
  }
}

void ADC_Proc(void)
{
  float fAdc_Val;

  if(ucSec != ucSec1)
  {
    ucSec1 = ucSec;

    uiAdc_Val = ADC1_Conv();
    fAdc_Val = (float)uiAdc_Val*3.3/4095;
    sprintf((char *)pucStr, "    %04u %4.2fV", uiAdc_Val, fAdc_Val);
    LCD_DisplayStringLine(Line5, pucStr);
    printf("%s", pucStr);

    uiAdc_Val = ADC1_InjeConv();
    fAdc_Val = (float)25+(5855.85-3.3*uiAdc_Val)/17.6085;
    sprintf((char *)pucStr, "    %04u %5.2f", uiAdc_Val, fAdc_Val);
    LCD_DisplayStringLine(Line6, pucStr);
    printf("%s\r\n", pucStr);
  }
}
// SysTick 中断处理程序
void SysTick_Handler(void)
{
  ulTick_ms++;
  if(ulTick_ms%1000 == 0) ucSec++;
}
```

ADC 范例程序的实现在 AT24C02 范例程序实现的基础上修改完成，步骤如下：

（1）将“205_AT24C02”文件夹复制并重命名为“206_ADC”文件夹，在“206_ADC”文件夹中双击工程文件“STM32.uvproj”打开工程。

（2）在“Inc”文件夹中新建文件“adc.h”。



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

（3）在“Sre”文件夹中新建文件“adc.c”。

（4）将下列库文件添加到“Library”组：

● stm32f10x_adc.c

（5）将“adc.c”添加到“Driver”组。

（6）修改文件“main.c”。

下载并运行程序，LCD 和 PC 超级终端上显示两个通道的转换值以及对应的电压值和温度值。

将实训平台上的电位器 R37 顺时针旋转到头，通道 8 的转换值和电压值分别为 0000 和 0.00V，温度值在 33.00℃左右。

将 R37 逆时针旋转到头，通道 8 的转换值和电压值分别为 4095V 和 3.30V，温度值升高到 35.00℃左右。

分析发现：温度值升高的原因是通道 16 的采样时间有问题。

默认状态下，ADCCLK 为 36MHz，此时的采样时间为 (239.5 + 12.5) / 36 = 7μs < 17.1μs。

将 ADCCLK 调整为 12MHz，此时的采样时间为 (239.5 + 12.5) / 12 = 21μs > 17.1μs，旋转电位器时温度值基本稳定。

## 5.7 TIM 应用程序设计

TIM 除了可以实现基本的定时功能外，还可以实现输入捕获和输出比较等功能。输入捕获可以测量 PWM 信号的周期和脉宽，输出比较可以输出单脉冲，也可以输出 PWM 信号。

TIM 使用的 GPIO 引脚如表 5.4 所示。

表 5.4 TIM 使用的 GPIO 引脚

| 定时器引脚 | GPIO 引脚 | | | | |
|---|---|---|---|---|---|
| | TIM1 | TIM2 | TIM3 | TIM4 | 配置 |
| CH1 | PA8 | PA0(1)（PA15）(2) | PA6（PB4/PC6）(2) | PB6 | 浮空输入（输入捕获）复用推挽输出（输出比较） |
| CH2 | PA9 | PA1（PB3）(2) | PA7（PB5/PC7）(2) | PB7 | |
| CH3 | PA10 | PA2（PB10）(2) | PB0（PC8）(2) | PB8 | |
| CH4 | PA11 | PA3（PB11）(2) | PB1（PC9）(2) | PB9 | |
| ETR | PA12 | PA0(1) | PD2 | - | 浮空输入 |
| BKIN | PB12（PA6）(2) | - | - | - | 浮空输入 |
| CH1N | PB13（PA7）(2) | - | - | - | 复用推挽输出 |
| CH2N | PB14（PB0）(2) | - | - | - | |
| CH3N | PB15（PB1）(2) | - | - | - | |

注：（1）TIM2_CH1 和 TIM2_ETR 共用一个引脚，但不能同时使用。

（2）括号中的引脚为复用功能重映射引脚。

TIM1～4 的时钟频率 TIMCLK 是 72MHz（时钟周期是 13.89ns），经过 16 位预分频后的最低频率和最长周期分别是：

72MHz / 65536 = 1099Hz，65536 / 72MHz = 910.22μs

再经过 16 位计数器分频后的最低频率和最长周期分别是：

1099Hz / 65536 = 16.77mHz，65536 / 1099Hz = 59.63ms

TIM 初始化时的主要工作是确定预分频值和周期值，对于单个通道，预分频值和周期值分别

电子工业出版社 PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

是（分频值 = 时钟频率 / 输出频率）：

$$\text{int}(\text{分频值} / 65536) < \text{预分频值} \leqslant \min(\text{分频值} / 2, 65536)$$

$$\text{周期值} = \text{分频值} / \text{预分频值}$$

例如输出频率为 10kHz 时，分频值为 7200，预分频值和周期值分别是：

$$0 < \text{预分频值} \leqslant 3600$$

$$\text{周期值} = 7200 / \text{预分频值}$$

当预分频值为 1 时，周期值为 7200；当预分频值为 3600 时，周期值为 2。

如果输出频率为 1kHz，则分频值为 72000，预分频值和周期值分别是：

$$1 < \text{预分频值} \leqslant 36000$$

$$\text{周期值} = 72000 / \text{预分频值}$$

当预分频值为 2 时，周期值为 36000；当预分频值为 36000 时，周期值为 2。

对于输出不同频率的多个通道，预分频值的最小值应该选取最低输出频率时的值，最大值应该选取最高输出频率时的值。

例如两个通道的输出频率分别为 1kHz 和 10kHz，当预分频值为 2 时，1kHz 的周期值为 36000，10kHz 的周期值为 3600；当预分频值为 3600 时，1kHz 的周期值为 20，10kHz 的周期值为 2。为了兼顾定时精度和灵敏度，预分频值可选择中间值（例如 72）。

用 TIM1 通道 2（PA9）输出 1kHz、占空比为 25%的矩形波，并用 TIM2 通道 2（PA1）测量矩形波周期和脉冲宽度的初始化程序如下：

TIM 头文件“tim.h”内容如下：

```
#include "stm32f10x.h"

void TIM1_OCInit(void);
void TIM2_ICInit(void);
unsigned char TIM2_Cap(unsigned int* puiTim_Val);
```

TIM 源文件“tim.c”内容如下：

```
#include "tim.h"
// TIM1 通道 2（PA9）输出矩形波初始化子程序
void TIM1_OCInit(void)
{
  GPIO_InitTypeDef GPIO_InitStruct;
  TIM_TimeBaseInitTypeDef TIM_TimeBaseInitStruct;
  TIM_OCInitTypeDef  TIM_OCInitStruct;

  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_TIM1, ENABLE);

  GPIO_InitStruct.GPIO_Pin = GPIO_Pin_9;
  GPIO_InitStruct.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_InitStruct.GPIO_Mode = GPIO_Mode_AF_PP;
  GPIO_Init(GPIOA, &GPIO_InitStruct);

  TIM_TimeBaseInitStruct.TIM_Prescaler = 71;
  TIM_TimeBaseInitStruct.TIM_CounterMode = TIM_CounterMode_Up;
```



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
  TIM_TimeBaseInitStruct.TIM_Period = 999;
  TIM_TimeBaseInitStruct.TIM_ClockDivision = TIM_CKD_DIV1;
  TIM_TimeBaseInitStruct.TIM_RepetitionCounter = 0;
  TIM_TimeBaseInit(TIM1, &TIM_TimeBaseInitStruct);

  TIM_OCInitStruct.TIM_OCMode = TIM_OCMode_PWM1;
  TIM_OCInitStruct.TIM_OutputState = TIM_OutputState_Enable;
  TIM_OCInitStruct.TIM_OutputNState = TIM_OutputNState_Disable;
  TIM_OCInitStruct.TIM_Pulse = 250;
  TIM_OCInitStruct.TIM_OCPolarity = TIM_OCPolarity_High;
  TIM_OC2Init(TIM1, &TIM_OCInitStruct);

  TIM_CtrlPWMOutputs(TIM1, ENABLE);
  TIM_Cmd(TIM1, ENABLE);
}
// TIM2 通道 2（PA1）测量矩形波周期和脉冲宽度初始化子程序
void TIM2_ICInit(void)
{
  GPIO_InitTypeDef GPIO_InitStruct;
  TIM_TimeBaseInitTypeDef TIM_TimeBaseInitStruct;
  TIM_ICInitTypeDef  TIM_ICInitStruct;

  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
  RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2, ENABLE);

  GPIO_InitStruct.GPIO_Pin = GPIO_Pin_1;
  GPIO_InitStruct.GPIO_Mode = GPIO_Mode_IN_FLOATING;
  GPIO_Init(GPIOA, &GPIO_InitStruct);

  TIM_TimeBaseInitStruct.TIM_Prescaler = 71;
  TIM_TimeBaseInitStruct.TIM_CounterMode = TIM_CounterMode_Up;
  TIM_TimeBaseInitStruct.TIM_Period = 65535;
  TIM_TimeBaseInitStruct.TIM_ClockDivision = TIM_CKD_DIV1;
  TIM_TimeBaseInit(TIM2, &TIM_TimeBaseInitStruct);

  TIM_ICInitStruct.TIM_Channel = TIM_Channel_2;
  TIM_ICInitStruct.TIM_ICPolarity = TIM_ICPolarity_Rising;
  TIM_ICInitStruct.TIM_ICSelection = TIM_ICSelection_DirectTI;
  TIM_ICInitStruct.TIM_ICPrescaler = TIM_ICPSC_DIV1;
  TIM_ICInitStruct.TIM_ICFilter = 0;
  TIM_PWMIConfig(TIM2, &TIM_ICInitStruct);

  TIM_SelectInputTrigger(TIM2, TIM_TS_TI2FP2);
  TIM_SelectSlaveMode(TIM2, TIM_SlaveMode_Reset);
  TIM_Cmd(TIM2, ENABLE);
}
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
// TIM2 测量方波周期和脉冲宽度处理子程序
unsigned char TIM2_Cap(unsigned int* puiTim_Val)
{
  if(TIM_GetFlagStatus(TIM2, TIM_FLAG_CC2))
  {
    puiTim_Val[0] = TIM_GetCapture2(TIM2)+1;
    puiTim_Val[1] = TIM_GetCapture1(TIM2)+1;
    return 1;
  }
  return 0;
}
```

TIM 范例程序主文件“main.c”内容如下：

```
#include "key.h"
#include "led.h"
#include "lcd.h"
#include "usart.h"
#include "i2c.h"
#include "adc.h"
#include "tim.h"

unsigned int  uiTim_Val[2];
unsigned char ucSec, ucSec1;
unsigned char pucStr[21];
unsigned long ulTick_ms;

void TIM_Proc(void);

int main(void)
{
  SysTick_Config(72000);          // 定时 1ms(HCLK = 72MHz)
  LED_Init();
  KEY_Init();

  STM3210B_LCD_Init();
  LCD_Clear(Blue);
  LCD_SetBackColor(Blue);
  LCD_SetTextColor(White);

  USART2_Init(9600);
  i2c_init();
  ADC1_Init();
  TIM1_OCInit();
  TIM2_ICInit();

  sprintf((char *)pucStr, "     TIM DEMO");
```



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
  LCD_DisplayStringLine(Line3, pucStr);
  printf("%s\r\n", pucStr);

  while(1)
  {
    LED_Disp(ucSec);
    TIM_Proc();
  }
}

void TIM_Proc(void)
{
  if(ucSec != ucSec1)
  {
    ucSec1 = ucSec;

    TIM2_Cap(uiTim_Val);

    sprintf((char *)pucStr, "     %4u %1ums ",
      uiTim_Val[0], uiTim_Val[0]/1000);
    LCD_DisplayStringLine(Line5, pucStr);
    printf("%3u %s", ucSec, pucStr);

    sprintf((char *)pucStr, "     %4u %2u%% ",
      uiTim_Val[1], uiTim_Val[1]*100/uiTim_Val[0]);
    LCD_DisplayStringLine(Line6, pucStr);
    printf("%s\r\n", pucStr);
  }
}
// SysTick 中断处理程序
void SysTick_Handler(void)
{
  ulTick_ms++;
  if(ulTick_ms%1000 == 0) ucSec++;
}
```

TIM 范例程序的实现在 ADC 范例程序实现的基础上修改完成，步骤如下：

（1）将“206_ADC”文件夹复制并重命名为“207_TIM”文件夹，在“207_TIM”文件夹中双击工程文件“STM32.uvproj”打开工程。

（2）在“Inc”文件夹中新建文件“tim.h”。

（3）在“Sre”文件夹中新建文件“tim.c”。

（4）将下列库文件添加到“Library”组：

- stm32f10x_tim.c

（5）将文件“tim.c”添加到“Driver”组。

（6）修改文件“main.c”。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

**注意**：运行程序时，连接 J1.6（PA9）和 J3.4（PA1）[小心：不要连接到 J3.3（+5V）上]，LCD 和超级终端上显示周期（1000 1ms）和脉冲宽度（250 25%）。

## 5.8 数码管静态显示程序设计

数码管由 8 个 LED 构成，其中 7 个 LED 组成数码显示，1 个 LED 作为小数点显示。通常将 8 个 LED 的其中一端连接在一起，根据连接在一起的引脚不同，数码管分为共阴极和共阳极两种。

数码管的显示方法有静态显示和动态显示两种。静态显示时每个数码管都一直显示，显示稳定，但硬件开销大；动态显示时每个数码管轮流显示，当每个数码管的显示时间小于一定值时，所有数码管看起来像“同时”显示。动态显示硬件开销小，但操作较复杂。

嵌入式竞赛扩展板上的 3 位数码管是共阴极静态显示，使用带输出锁存的 8 位移位寄存器 74LS595 驱动，74LS595 的引脚功能如表 5.5 所示。

表 5.5　74LS595 的引脚功能

| 引脚名称 | 引脚方向 | 引脚功能 | 引脚名称 | 引脚方向 | 引脚功能 |
|---|---|---|---|---|---|
| SER | 输入 | 串行数据 | /OE | 输入 | 锁存输出允许 |
| SRCLK | 输入 | 移位寄存器时钟↑ | QA~QH | 输出 | 8 位并行数据 |
| /SRCLR | 输入 | 移位寄存器清零 | QH’ | 输出 | 串行数据（级联用） |
| RCLK | 输入 | 输出锁存时钟↑ | | | |

通过 GPIO 接口输出显示数据的硬件连接如下：

- P4.1（PA1）-P3.1（SER）
- P4.2（PA2）-P3.2（RCK）
- P4.3（PA3）-P3.3（SCK）

同时断开实训平台上的下列连接：

- J1.3（PA3）-J2.3（RXD2）
- J1.4（PA2）-J2.4（TXD2）

**注意**：由于数码管和 USART2 都使用了 PA2 和 PA3 引脚，所以数码管和 USART2 不能同时使用。

数码管头文件“seg.h”内容如下：

```
#include "stm32f10x.h"

void SEG_Init(void);
void SEG_Disp(unsigned char ucData1, unsigned char ucData2,
  unsigned char ucData3, unsigned char ucDot);
```

数码管源文件“seg.c”内容如下：

```
#include "seg.h"
// SEG 接口初始化
void SEG_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct;
  // 允许 GPIOA 时钟
```



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
  // PA1-SER、PA2-RCK 和 PA3-SCK 通用推挽输出
  GPIO_InitStruct.GPIO_Pin = GPIO_Pin_1 | GPIO_Pin_2 | GPIO_Pin_3;
  GPIO_InitStruct.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_InitStruct.GPIO_Mode = GPIO_Mode_Out_PP;
  GPIO_Init(GPIOA, &GPIO_InitStruct);
}
// SEG 显示
// 入口参数：ucData1、ucData2 和 ucData3-3 个显示数据，ucDot-3 个小数点
void SEG_Disp(unsigned char ucData1, unsigned char ucData2,
  unsigned char ucData3, unsigned char ucDot)
{
  unsigned char i;
  unsigned char code[17]={0x3f, 0x06, 0x5b, 0x4f, 0x66, 0x6d, 0x7d, 0x07,
    0x7f, 0x6f, 0x77, 0x7c, 0x39, 0x5e, 0x79, 0x71, 0x00};
  unsigned long ulData = (code[ucData3] << 16) + (code[ucData2] << 8)
    + code[ucData1];

  ulData += (ucDot&1)<<23;
  ulData += (ucDot&2)<<14;
  ulData += (ucDot&4)<<5;

  GPIO_ResetBits(GPIOA, GPIO_Pin_2);              // PA2(RCK)=0
  for(i=0; i<24; i++)
  {
    GPIO_ResetBits(GPIOA, GPIO_Pin_3);            // PA3(SCK)=0
    if(ulData & 0x800000)                         // 从高位开始发送
      GPIO_SetBits(GPIOA, GPIO_Pin_1);            // PA1(SER)=1
    else
      GPIO_ResetBits(GPIOA, GPIO_Pin_1);          // PA1(SER)=0
    ulData <<= 1;
    GPIO_SetBits(GPIOA, GPIO_Pin_3);              // PA3(SCK)=1
  }
  GPIO_SetBits(GPIOA, GPIO_Pin_2);                // PA2(RCK)=1
}
```

数码管范例程序主文件“main.c”内容如下：

```
#include "key.h"
#include "led.h"
#include "lcd.h"
#include "seg.h"

unsigned int  uiSeg;
unsigned char ucSec, ucSec1, ucDot;
unsigned long ulTick_ms;
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
int main(void)
{
  SysTick_Config(72000);          // 定时 1ms(HCLK = 72MHz)
  KEY_Init();
  LED_Init();

  STM3210B_LCD_Init();
  LCD_Clear(Blue);
  LCD_SetBackColor(Blue);
  LCD_SetTextColor(White);

  SEG_Init();
  SEG_Disp(0x10, 0x10, 0x10, 0); // 熄灭所有显示

  while(1)
  {
    if(ucSec1 != ucSec)
    {
      ucSec1 = ucSec;

      SEG_Disp((uiSeg&0xf00)>>8, (uiSeg&0xf0)>>4, uiSeg&0xf, ucDot++);
      uiSeg += 0x111;
      if(uiSeg > 0x1000) uiSeg = 0;
    }
  }
}

// SysTick 中断处理程序
void SysTick_Handler(void)
{
  ulTick_ms++;
  if(ulTick_ms%1000 == 0) ucSec++;
}
```

数码管范例程序的实现可以在 LCD 范例程序实现的基础上修改完成。

## 5.9 ADC 键盘扫描程序设计

扩展板上的按键通过电阻分压 ADC 转换进行识别，各按键对应的电阻值和转换值如表 5.6 所示［转换值=4095×电阻值/(10000+电阻值)］。

表 5.6 各按键对应的电阻值和转换值

| 按键 | S1 | S2 | S3 | S4 | S5 | S6 | S7 | S8 |
|---|---|---|---|---|---|---|---|---|
| 电阻值 | 0 | 510 | 1710 | 3310 | 5110 | 8110 | 12810 | 22810 |
| 转换值 | 0 | 198 | 597 | 1018 | 1384 | 1833 | 2299 | 2846 |



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

**注意**：表中的转换值是理论计算值，扩展板各按键的实际转换值可能与表中不同，具体转换值可通过程序中的相关语句获取。

通过 ADC 识别按键的硬件连接如下：

- P4.5（PA5）-P5.5（AKEY）

ADC 按键程序设计在 ADC 程序设计的基础上修改完成，步骤如下：

（1）在“Inc”文件夹的“adc.h”中添加下列函数声明：

```
void ADC1_Init_AKEY(void);
unsigned char AKEY_Read(void);
void Delay_AKEY(unsigned int ms);
```

（2）在“Sre”文件夹的“adc.c”中添加下列函数代码：

```
void ADC1_Init_AKEY(void)
{
  GPIO_InitTypeDef GPIO_InitStruct;
  ADC_InitTypeDef ADC_InitStruct;
  // 允许 GPIOA 和 ADC1 时钟
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1, ENABLE);
  // PA5-IN5 模拟输入
  GPIO_InitStruct.GPIO_Pin = GPIO_Pin_5;
  GPIO_InitStruct.GPIO_Mode = GPIO_Mode_AIN;
  GPIO_Init(GPIOA, &GPIO_InitStruct);
  // 初始化 ADC1
  ADC_InitStruct.ADC_Mode = ADC_Mode_Independent;
  ADC_InitStruct.ADC_ScanConvMode = DISABLE;
  ADC_InitStruct.ADC_ContinuousConvMode = DISABLE;
  ADC_InitStruct.ADC_ExternalTrigConv = ADC_ExternalTrigConv_None;
  ADC_InitStruct.ADC_DataAlign = ADC_DataAlign_Right;
  ADC_InitStruct.ADC_NbrOfChannel = 1;
  ADC_Init(ADC1, &ADC_InitStruct);
  // 配置通道 5
  ADC_RegularChannelConfig(ADC1, ADC_Channel_5, 1,
    ADC_SampleTime_1Cycles5);
  // 启动 ADC1
  ADC_Cmd(ADC1, ENABLE);
  // 校准 ADC1
  ADC_StartCalibration(ADC1);
  while(ADC_GetCalibrationStatus(ADC1));
}

unsigned char AKEY_Read(void)
{
  unsigned char ucAkey_Val = 0;
  unsigned int  uiAdc_Val = ADC1_Conv();
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
  if(uiAdc_Val < 3470)                // 按键按下
  {
    Delay_AKEY(100);                  // 延时消抖
    uiAdc_Val = ADC1_Conv();
    if(uiAdc_Val < 3470)              // (4095+2846)/2=3470
    {
      if(uiAdc_Val > 2572)            // (2846+2299)/2=2572
        ucAkey_Val = 8;
      else if(uiAdc_Val > 2066)       // (2299+1833)/2=2066
        ucAkey_Val = 7;
      else if(uiAdc_Val > 1608)       // (1833+1384)/2=1608
        ucAkey_Val = 6;
      else if(uiAdc_Val > 1201)       // (1384+1018)/2=1201
        ucAkey_Val = 5;
      else if(uiAdc_Val > 807)        // (1018+597)/2=807
        ucAkey_Val = 4;
      else if(uiAdc_Val > 498)        // (597+198)/2=498
        ucAkey_Val = 3;
      else if(uiAdc_Val > 99)         // 198/2=99
        ucAkey_Val = 2;
      else
        ucAkey_Val = 1;
    }
  }
  return ucAkey_Val;
}

void Delay_AKEY(unsigned int ms)
{
  unsigned int i, j;

  for(i=0; i<ms; i++)
    for(j=0; j<7992; j++);            // SYSCLK = 72MHz
//  for(j=0; j<1598; j++);            // SYSCLK = 8MHz
}
```

ADC 按键范例程序主文件“main.c”内容如下：

```
#include "key.h"
#include "led.h"
#include "lcd.h"
#include "seg.h"
#include "adc.h"

unsigned int  uiSeg;
unsigned char ucDot, ucAkey_Long;
unsigned long ulTick_ms, ulAkey_Time;
```



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
void AKEY_Proc(void);

int main(void)
{
  SysTick_Config(72000);          // 定时 1ms(HCLK = 72MHz)
  KEY_Init();
  LED_Init();

  STM3210B_LCD_Init();
  LCD_Clear(Blue);
  LCD_SetBackColor(Blue);
  LCD_SetTextColor(White);

  ADC1_Init_AKEY();
  SEG_Init();
  SEG_Disp(0, 0, 0, 0);

  while(1)
  {
    AKEY_Proc();
    SEG_Disp((uiSeg&0xf00)>>8, (uiSeg&0xf0)>>4, uiSeg&0xf, ucDot);
  }
}

void AKEY_Proc(void)
{
  unsigned char ucAkey_Val;
  ucAkey_Val = AKEY_Read();
  if(ucAkey_Val != ucAkey_Long)
  {
    ucAkey_Long = ucAkey_Val;
    ulAkey_Time = ulTick_ms;
  }
  else
    ucAkey_Val = 0;

  if(ucAkey_Val)
  {
    uiSeg = (uiSeg << 4) + ucAkey_Val;
    ucDot++;
  }
}
// SysTick 中断处理程序
void SysTick_Handler(void)
{
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
    ulTick_ms++;
  }
```

ADC 按键范例程序的实现可以在数码管范例程序实现的基础上修改完成。

## 5.10 DHT11 湿度传感器程序设计

DHT11 是单线接口数字温、湿度传感器，温度测量范围是 0～50℃，湿度测量范围是 20%～90%RH，温度测量精度是±2℃，湿度测量精度是±5%RH。

DHT11 包含一个电阻式感湿元件和一个 NTC（负温度系数）测温元件，通过双向单线输出温、湿度数据，一次数据输出为 40 位（高位在前，大约需要 4ms），数据格式为：

8 位湿度整数 ＋8 位湿度小数（0）＋8 位温度整数 ＋8 位温度小数（0）＋8 位校验和其中校验和是前 4 个 8 位数据之和的后 8 位。

MCU 通过单线读取 DHT11 输出数据过程如图 5.2 所示。

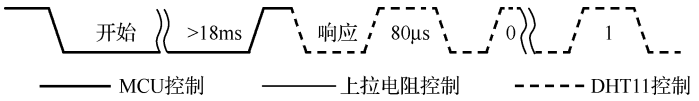



图 5.2　MCU 通过单线读取 DHT11 输出数据过程

单线空闲时为高电平，MCU 读取数据时首先发送开始信号（输出低电平，持续时间必须大于 18ms，以保证 DHT11 能检测到开始信号），然后切换到输入模式（单线由上拉电阻拉为高电平）等待 DHT11 响应。

DHT11 检测到开始信号后触发一次数据采集，并等待单线变为高电平后输出响应信号（低电平，持续时间 80μs），然后输出高电平（持续时间 80μs）准备输出 40 位数据。每位数据都以低电平（持续时间 50μs）开始，输出 0 时高电平持续时间为 26～28μs，输出 1 时高电平持续时间为 70μs。最后一位数据输出后输出低电平（持续时间 50μs），单线由上拉电阻拉为高电平进入空闲状态。

MCU 检测到响应信号后从单线读取 40 位数据，并判断校验和是否正确，如果正确则数据有效，否则丢弃数据。

使用 DHT11 的硬件连接如下：

- P4.7（PA7）-P3.7（HDQ）

DHT11 程序设计在竞赛资源包“底层驱动代码参考”文件夹中“dht11.h”和“dht11.c”的基础上修改实现。

DHT11 范例程序主文件“main.c”内容如下：

```
#include "key.h"
#include "led.h"
#include "lcd.h"
#include "stdio.h"
#include "dht11.h"

unsigned int  uiDht_Val;
unsigned char ucSec, ucSec1;
unsigned char pucStr[21];
unsigned long ulTick_ms;
```



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
void DHT_Proc(void);

int main(void)
{
  SysTick_Config(72000);          // 定时 1ms(HCLK = 72MHz)
  KEY_Init();
  LED_Init();

  STM3210B_LCD_Init();
  LCD_Clear(Blue);
  LCD_SetBackColor(Blue);
  LCD_SetTextColor(White);

  dht11_init();

  while(1)
  {
    LED_Disp(ucSec);
    DHT_Proc();
  }
}

void DHT_Proc(void)
{
  if(ucSec != ucSec1)
  {
    ucSec1 = ucSec;

    uiDht_Val = dht11_read();
    sprintf((char *)pucStr, "  Humidity:   %2d%%", uiDht_Val>>24);
    LCD_DisplayStringLine(Line3, pucStr);
    sprintf((char *)pucStr, "  Temperature: %2dC", (uiDht_Val>>8)&0xff);
    LCD_DisplayStringLine(Line5, pucStr);
  }
}
// SysTick 中断处理程序
void SysTick_Handler(void)
{
  ulTick_ms++;
  if(ulTick_ms%1000 == 0) ucSec++;
}
```

DHT11 范例程序的实现可以在 LCD 范例程序实现的基础上修改完成。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

## 5.11 DS18B20 温度传感器程序设计

DS18B20 是单线接口数字温度传感器，测量范围是−55℃～+125℃，−10℃～+85℃范围内精度是±0.5℃，测量分辨率为 9～12 位（复位值为 12 位，最大转换时间为 750ms）。

DS18B20 包括寄生电源电路、64 位 ROM 和单线接口电路、暂存器、EEPROM、8 位 CRC 生成器和温度传感器等。寄生电源电路可以实现外部电源供电和单线寄生供电，64 位 ROM 中存放的 48 位序列号用于识别同一单线上连接的多个 DS18B20，以实现多点测温。

64 位 ROM 代码的格式为：

8 位 CRC 校验码 ＋48 位序列号 ＋8 位系列码（0x28）

其中 8 位 CRC 校验码是 48 位序列号和 8 位系列码的 CRC 校验码。

DS18B20 的暂存器如表 5.7 所示。

表 5.7 DS18B20 的暂存器

| 地　址 | 名　称 | 类　型 | 复 位 值 | 说　明 |
|---|---|---|---|---|
| 0 | 温度值低 8 位 | 只读 | 0x0550<br>（85℃） | b15～b11：符号位，b10～b4：7 位整数<br>b3～b0：4 位小数（补码） |
| 1 | 温度值高 8 位 | 只读 | | |
| 2 | TH 或用户字节 1 | 读写 | EEPROM | b7：符号位，b6～b0：7 位温度报警高值（补码） |
| 3 | TL 或用户字节 2 | 读写 | EEPROM | b7：符号位，b6～b0：7 位温度报警低值（补码） |
| 4 | 配置寄存器 CR | 读写 | EEPROM | b6～b5：分辨率，00～11：9~12 位 |
| 5 | 保留 | 只读 | 0xFF | |
| 6 | 保留 | 只读 | 0x0C | |
| 7 | 保留 | 只读 | 0x10 | |
| 8 | CRC | 只读 | EEPROM | 暂存器 0～7 数据 CRC 校验码 |

DS18B20 的操作包括下列 3 步：

- 复位
- ROM 命令
- 功能命令

ROM 命令和功能命令分别如表 5.8 和表 5.9 所示。

表 5.8 ROM 命令

| 命　令 | 代　码 | 参数或返回值 | 说　明 |
|---|---|---|---|
| 搜索 ROM | 0xF0 | – | 搜索单线上连接的多个 DS18B20，搜索后重新初始化 |
| 读取 ROM | 0x33 | ROM 代码 | 读取单个 DS18B20 的 64 位 ROM 代码 |
| 匹配 ROM | 0x55 | ROM 代码 | 寻址指定 ROM 代码的 DS18B20 |
| 跳过 ROM | 0xCC | – | 寻址所有单线上连接的多个 DS18B20 |
| 搜索报警 | 0xEC | | 搜索单线上连接的有报警标志的 DS18B20 |

表 5.9 功 能 命 令

| 命　令 | 代　码 | 参数或返回值 | 说　明 |
|---|---|---|---|
| 转换温度 | 0x44 | 0—转换，1—完成 | 启动温度转换，转换结果存放在暂存器的 0～1 字节 |



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

续表

| 命　令 | 代　码 | 参数或返回值 | 说　明 |
|---|---|---|---|
| 读暂存器 | 0xBE | 9 字节数据 | 读取暂存器的 0～8 字节 |
| 写暂存器 | 0x4E | TH TL CR | 将 TH、TL 和 CR 值写入暂存器的 2～4 字节 |
| 复制暂存器 | 0x48 | – | 将暂存器的 2～4 字节复制到 EEPROM |
| 调回 EEPROM | 0xB8 | 0—调回，1—完成 | 将 EEPROM 的值调回到暂存器的 2～4 字节 |
| 读电源模式 | 0xB4 | – | 确定 DS18B20 是否使用寄生供电模式 |

使用 DS18B20 的硬件连接如下：

- P4.6（PA6）-P3.6（TDQ）

DS18B20 程序设计在竞赛资源包“底层驱动代码参考”文件夹中“ds18b20.h”和“ds18b20.c”的基础上修改实现：

在“ds18b20.h”中追加下列声明：

```
unsigned int dsb_read(void);
```

在“ds18b20.c”中追加下列内容：

```
unsigned int dsb_read(void)
{
  unsigned char th, tl;

  ow_reset();                          // 复位
  ow_byte_wr(0xCC);                    // 跳过 ROM
  ow_byte_wr(0x44);                    // 转换温度

  ow_reset();
  ow_byte_wr(0xCC);
  ow_byte_wr(0xBE);                    // 读暂存器
  tl = ow_byte_wr(0xFF);               // 读温度值低 8 位
  th = ow_byte_wr(0xFF);               // 读温度值高 8 位
  return (th<<8) + tl;
}
```

DS18B20 范例程序主文件“main.c”内容如下：

```
#include "key.h"
#include "led.h"
#include "lcd.h"
#include "stdio.h"
#include "ds18b20.h"

unsigned int  uiDsb_Val;
unsigned char ucSec, ucSec1;
unsigned char pucStr[21];
unsigned long ulTick_ms;
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
void DSB_Proc(void);

int main(void)
{
  SysTick_Config(72000);              // 定时 1ms(HCLK = 72MHz)
  KEY_Init();
  LED_Init();

  STM3210B_LCD_Init();
  LCD_Clear(Blue);
  LCD_SetBackColor(Blue);
  LCD_SetTextColor(White);

  ds18b20_init_x();

  while(1)
  {
    LED_Disp(ucSec);
    DSB_Proc();
  }
}

void DSB_Proc(void)
{
  if(ucSec != ucSec1)
  {
    ucSec1 = ucSec;

    uiDsb_Val = dsb_read();
    sprintf((char *)pucStr, "    Temp: %5.2fC", uiDsb_Val/16.0);
    LCD_DisplayStringLine(Line5, pucStr);
  }
}
// SysTick 中断处理程序
void SysTick_Handler(void)
{
  ulTick_ms++;
  if(ulTick_ms%1000 == 0) ucSec++;
}
```

DS18B20 范例程序的实现可以在 LCD 范例程序实现的基础上修改完成。

## 5.12 LIS302DL 加速度传感器程序设计

LIS302DL 是三轴±2*g*/±8*g* 智能数字输出加速度传感器，提供 I2C/SPI 接口，扩展板使用的是 I2C 接口，器件地址是 0x38。



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

LIS302DL 主要内部寄存器如表 5.10 所示。

表 5.10　LIS302DL 主要内部寄存器

| 地　址 | 名　称 | 类　型 | 复 位 值 | 说　明 |
|---|---|---|---|---|
| 0x0F | ID | 只读 | 0x3B | 标识寄存器 |
| 0x20 | Ctrl_Reg1 | 读写 | 0x07 | 控制寄存器 1（b6-PD：掉电控制，1—正常模式） |
| 0x21 | Ctrl_Reg2 | 读写 | 0x00 | 控制寄存器 2 |
| 0x22 | Ctrl_Reg3 | 读写 | 0x00 | 控制寄存器 3（中断控制寄存器） |
| 0x27 | Status_Reg | 只读 | 0x00 | 状态寄存器（b7～b4：溢出标志，b3～b0：更新标志） |
| 0x29 | OutX | 只读 | – | X 轴输出寄存器 |
| 0x2B | OutY | 只读 | – | Y 轴输出寄存器 |
| 0x2D | OutZ | 只读 | – | Z 轴输出寄存器 |

使用加速度传感器的硬件连接如下：

- P2.1（PA4）-P2.2（SCL）
- P2.3（PA5）-P2.4（SDA）

同时断开下列连接：

- P4.4（PA4）-P3.4（AO1）或 P5.4（TRAO）
- P4.5（PA5）-P3.5（AO2）或 P5.5（AKEY）

**注意**：由于引脚冲突，加速度传感器不能和 AO1/AO2 或 TRAO/AKEY 同时使用。

加速度传感器程序设计可以在 AT24C02 程序设计的基础上修改完成，步骤如下：

（1）在“D:\CT117E\Inc”文件夹中将文件“i2c.h（AT24C02 头文件）”复制并重命名为“lis302.h（LIS302 头文件）”。

（2）在“D:\CT117E\Sre”文件夹中将文件“i2c.c（AT24C02 库文件）”复制并重命名为“lis302.c（LIS302 库文件）”。

（3）将“lis302.c”中的引脚定义

```
#define I2C_PORT GPIOB
#define SDA_Pin GPIO_Pin_7
#define SCL_Pin GPIO_Pin_6
```

修改为

```
#define I2C_PORT GPIOA
#define SDA_Pin GPIO_Pin_5
#define SCL_Pin GPIO_Pin_4
```

（4）将 i2c_read()函数中的器件地址 0xa0 和 0xa1 分别修改为 0x38 和 0x39。

（5）将 i2c_write()函数中的器件地址 0xa0 修改为 0x38。

LIS302 范例程序主文件“main.c”内容如下：

```
#include "key.h"
#include "led.h"
#include "lcd.h"
#include "stdio.h"
#include "lis302.h"
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
unsigned char pucBuf[1];
unsigned char ucSec, ucSec1;
unsigned char pucStr[21];
unsigned long ulTick_ms;

void LIS_Proc(void);

int main(void)
{
  SysTick_Config(72000);         // 定时 1ms(HCLK = 72MHz)
  KEY_Init();
  LED_Init();

  STM3210B_LCD_Init();
  LCD_Clear(Blue);
  LCD_SetBackColor(Blue);
  LCD_SetTextColor(White);

  i2c_init();
  pucBuf[0] = 0x47;
  i2c_write(pucBuf, 0x20, 1);    // 设置正常模式

  while(1)
  {
    LED_Disp(ucSec);
    LIS_Proc();
  }
}

void LIS_Proc(void)
{
  if(ucSec != ucSec1)
  {
    ucSec1 = ucSec;

    i2c_read(pucBuf, 0x29, 1);
    sprintf((char *)pucStr, "     OutX: %02X", pucBuf[0]);
    LCD_DisplayStringLine(Line4, pucStr);

    i2c_read(pucBuf, 0x2B, 1);
    sprintf((char *)pucStr, "     OutY: %02X", pucBuf[0]);
    LCD_DisplayStringLine(Line5, pucStr);

    i2c_read(pucBuf, 0x2D, 1);
    sprintf((char *)pucStr, "     OutZ: %02X", pucBuf[0]);
```



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
    LCD_DisplayStringLine(Line6, pucStr);
  }
}
// SysTick 中断处理程序
void SysTick_Handler(void)
{
  ulTick_ms++;
  if(ulTick_ms%1000 == 0) ucSec++;
}
```

LIS302 范例程序的实现可以在 LCD 范例程序实现的基础上修改完成。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

# 第 6 章　嵌入式设计与开发项目往届试题解析

本章介绍嵌入式设计与开发项目往届试题解析，包括系统设计、初始化程序设计、处理程序设计和系统实现等，系统设计包括方框图和主程序，初始化程序设计在模块程序设计的基础上实现，工程文件存放在“D:\CT117E”文件夹中。

## 6.1　第七届预赛试题——液位检测告警系统

液位检测告警系统通过采集模拟电压信号计算液位高度，并根据用户设置的液位阈值执行报警动作，在液位等级发生变化时，通过串行接口将液位信息发送到 PC。液位检测告警系统方框图如图 6.1 所示。

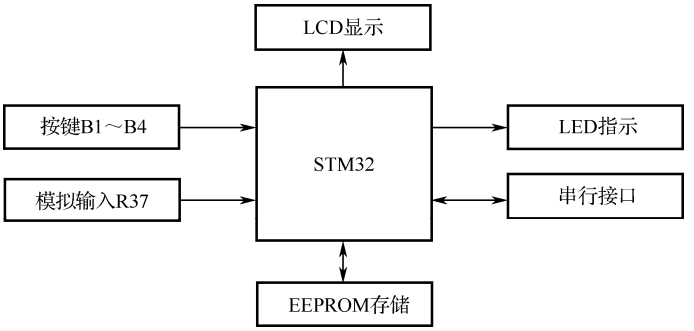



图 6.1　液位检测告警系统方框图

液位检测告警系统功能描述如下：

**1）液位检测**

通过电位器 R37 模拟液位传感器输出电压信号，系统以 1s 为间隔采集 R37，并与用户设定的液位阈值进行比较。

假定液位高度与 R37 输出电压之间具有正比例关系：

$$H = KV_{R37}$$

当 $V_{R37}$ = 3.3V 时，对应液位高度为 100cm。

通过 LCD 显示当前的液位高度、传感器（R37）输出状态和液位等级，液位检测显示界面如图 6.2 所示。

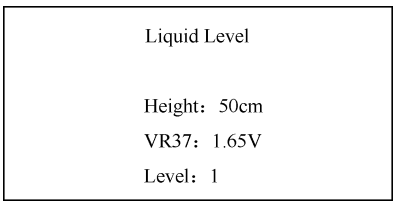



图 6.2　液位检测显示界面

**2）阈值设置**

系统可设置三个液位阈值，对应四个液位等级。阈值由用户通过按键设置并保存，系统根据设置阈值判断液位等级。

假定用户设置的三个液位阈值分别为 10cm、20cm 和 30cm，液位高度与液位等级的对应关系如下：

（1）液位高度≤10cm 时，液位等级为 0。

（2）10cm<液位高度≤20cm 時，液位等级为 1。

（3）20cm<液位高度≤30cm 时，液位等级为 2。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

（4）液位高度>30cm 时，液位等级为 3。

用户修改阈值后，系统应将阈值保存在 EEPROM 中，当系统重新上电时，可从 EEPROM 中读取，阈值设置显示界面如图 6.3 所示。

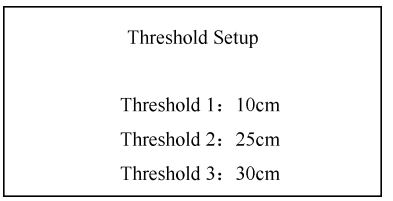



图 6.3　阈值设置显示界面

**3）按键功能**

系统使用 4 个按键：B1、B2、B3 和 B4。

按键 B1 为设置键，按下后进入阈值设置显示界面，再次按下时保存设置结果到 EEPROM，并返回液位检测显示界面。

按键 B2 为切换键，切换选择 3 个阈值，被选择的阈值突出显示。

按键 B3 为加键，每按一次，阈值增加 5cm，增加到 95cm 为止。

按键 B4 为减键，每按一次，阈值减少 5cm，减少到 5cm 为止。

**4）串口查询与输出功能**

（1）查询：通过 PC 向系统发送字符“C”，系统返回当前液位高度和液位等级；发送字符“S”，返回当前设置的三个阈值。串口波特率设置为 9600。

液位高度和等级返回数据格式举例如下：

C:H55+L2\r\n　　液位高度为 55cm，液位等级为 2

阈值返回数据格式举例如下：

S:TL30+TM50+TH70\r\n　　三个阈值分别为 30cm、50cm 和 70cm

液位高度≤10cm 时，液位等级为 0。

（2）输出：当液位等级发生变化时，系统向 PC 发送当前液位等级、液位高度和液位变化趋势（上升或下降）。

输出数据格式举例如下：

A:H55+L2+D\r\n　　液位高度 55cm，液位等级为 2，变化趋势下降

A:H55+L2+U\r\n　　液位高度 55cm，液位等级为 2，变化趋势上升

**5）状态指示**

LED 指示功能定义如下：

LD1：运行状态指示灯，以 1s 为间隔闪烁。

LD2：液位等级变化指示灯，当液位等级发生变化时，LD2 以 0.2s 为间隔闪烁 5 次。

LD3：通信状态指示灯，当系统接收到查询指令时，LD3 以 0.2s 为间隔闪烁 5 次。

### 6.1.1　系统设计

通过分析系统功能，可以得到系统详细方框图，如图 6.4 所示。

液位检测告警系统主程序流程图如图 6.5 所示。

主程序首先对系统进行初始化，包括 SysTick 初始化、按键接口初始化、LED 接口初始化、LCD 接口初始化、UART 接口初始化、I2C 接口初始化和 ADC 初始化等。系统初始化后从 AT24C02 中读取阈值。

主循环的操作包括按键处理、LCD 处理、ADC 处理和 UART 处理，按键处理中包含保存阈值，ADC 处理中包含 UART 发送，UART 处理中包含 UART 接收和发送，LED 处理在 SysTick 中断处理程序中实现。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

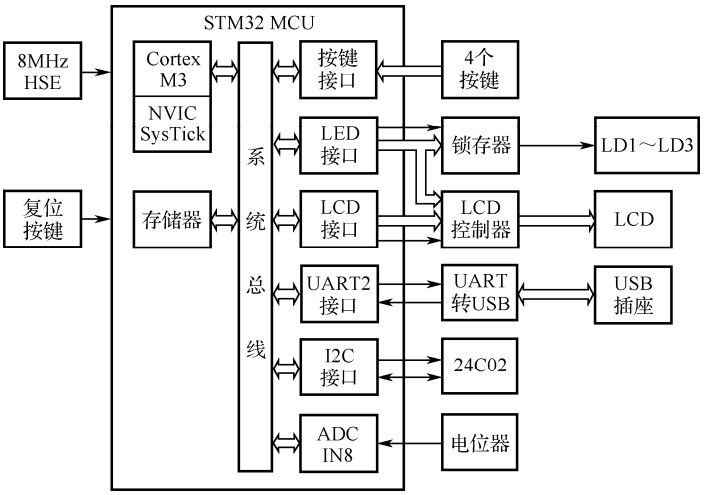



图 6.4　液位检测告警系统详细方框图

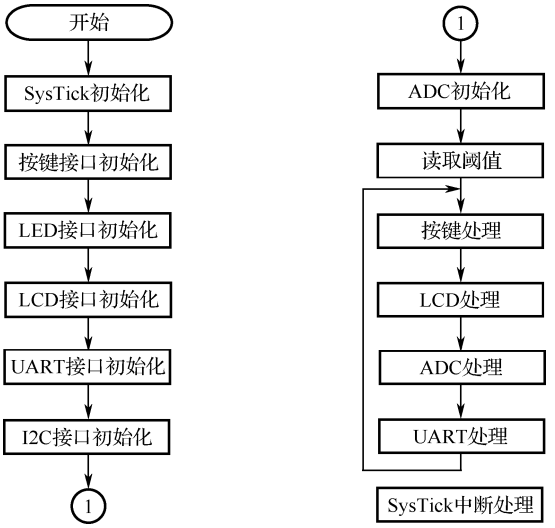



图 6.5　液位检测告警系统主程序流程图

主程序设计如下：

```
#include "key.h"
#include "led.h"
#include "lcd.h"
#include "usart.h"
#include "i2c.h"
#include "adc.h"

unsigned int uiAdc_Val;
unsigned char ucHeight, ucLevel, ucLevel1;
unsigned char ucState, pucStr[21], pucTh[3], pucRcv[1];
unsigned char ucLed, ucLd2, ucLd3, ucNum = 10;
unsigned char ucSec, ucSec1, ucKey_Long;
unsigned long ulTick_ms;
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
void KEY_Proc(void);
void LCD_Proc(void);
void ADC_Proc(void);
void UART_Proc(void);

int main(void)
{
  SysTick_Config(72000);
  KEY_Init();
  LED_Init();

  STM3210B_LCD_Init();
  LCD_Clear(Blue);
  LCD_SetBackColor(Blue);
  LCD_SetTextColor(White);

  USART2_Init(9600);
  i2c_init();
  ADC1_Init();

  i2c_read(pucTh, 0, 3);
  if(pucTh[0] > 100) pucTh[0] = 10;
  if(pucTh[1] > 100) pucTh[0] = 20;
  if(pucTh[2] > 100) pucTh[0] = 30;

  while(1)
  {
    KEY_Proc();
    LCD_Proc();
    ADC_Proc();
    UART_Proc();
  }
}
```

### 6.1.2 初始化程序设计

初始化程序包括 SysTick 初始化、按键接口初始化、LED 接口初始化、LCD 接口初始化、UART 接口初始化、I2C 接口初始化和 ADC 初始化等。

**1）SysTick 初始化程序设计**

SysTick 初始化用 SysTick_Config()实现。

**2）按键接口初始化程序设计**

按键接口初始化程序使用“key.c”中的 KEY_Init()实现。

**3）LED 接口初始化程序设计**

LED 接口初始化程序使用“led.c”中的 LED_Init()实现。

**4）LCD 接口初始化程序设计**

LCD 接口初始化程序使用“lcd.c”中的 STM3210B_LCD_Init()实现。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

5）UART 接口初始化程序设计

UART 接口初始化程序使用“usart.c”中的 USART2_Init()实现。

6）I2C 接口初始化程序设计

I2C 接口初始化程序使用“i2c.c”中的 i2c_init()实现。

7）ADC 初始化程序设计

ADC 初始化程序使用“adc.c”中的 ADC1_Init()实现。

### 6.1.3 处理程序设计

处理程序包括按键处理、LCD 处理、ADC 处理、UART 处理和 LED 处理等，LED 处理在 SysTick 中断处理程序中实现。液位检测告警系统处理程序流程图如图 6.6 所示。

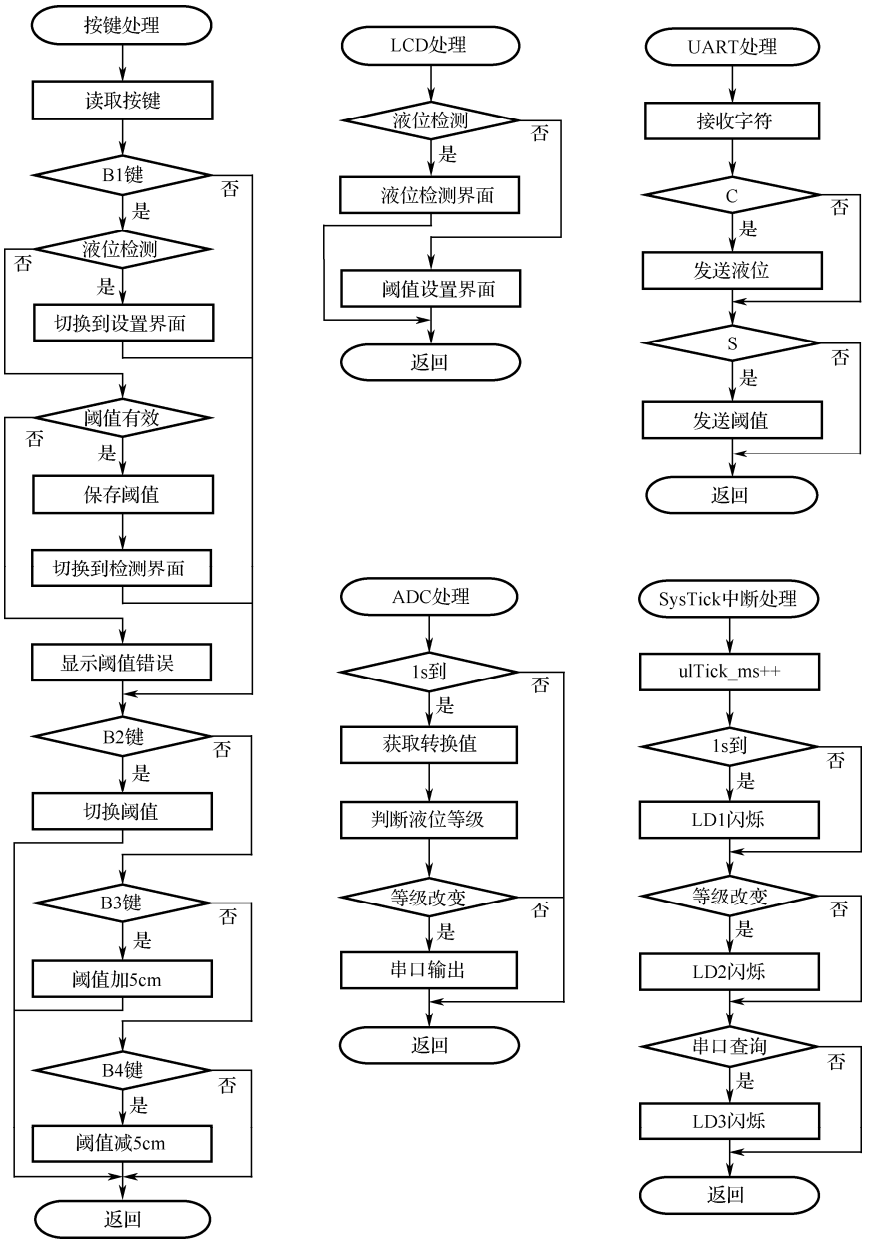



图 6.6 液位检测告警系统处理程序流程图



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

1）按键处理程序设计

按键处理程序根据状态图进行设计。液位检测告警系统工作状态图如图 6.7 所示。

系统共有 4 个状态：1 个液位检测状态和 3 个阈值状态，液位检测状态和阈值状态间用 B1 键切换，3 个阈值状态间用 B2 键切换，3 个阈值状态下用 B3 键或 B4 键加或减阈值。

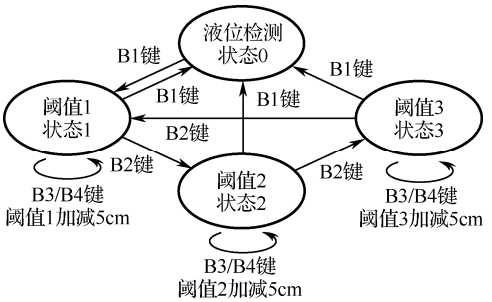



图 6.7　液位检测告警系统工作状态图

按键处理程序设计如下：

```
void KEY_Proc(void)
{
  unsigned char ucKey_Val;

  ucKey_Val = KEY_Scan();
  if(ucKey_Val != ucKey_Long)
    ucKey_Long = ucKey_Val;
  else
    ucKey_Val = 0;

  switch(ucKey_Val)
  {
    case 1:                                                    // B1: 设置键
      if(!ucState)
        ucState = 1;
      else
      {
        if((pucTh[0] < pucTh[1]) && (pucTh[1] < pucTh[2]))
        {
          LCD_DisplayStringLine(Line3, (u8*)"                  ");

          i2c_write(pucTh, 0, 3);                              // 保存阈值
          ucState = 0;
        }
        else
        {
          LCD_SetTextColor(Red);
          LCD_DisplayStringLine(Line3, (u8*)"  Threshold Error   ");
          LCD_SetTextColor(White);
        }
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
      }break;
    case 2:                                                    // B2: 切换键
      if(ucState)
        if(++ucState == 4)
          ucState = 1;
      break;
    case 3:                                                    // B3: 阈值加
      if(ucState)
        if(pucTh[ucState-1] < 95)
          pucTh[ucState-1] += 5;
      break;
    case 4:                                                    // B4: 阈值减
      if(ucState)
        if(pucTh[ucState-1] > 5)
          pucTh[ucState-1] -= 5;
  }
}
```

2）LCD 处理程序设计

LCD 处理程序根据 LCD 显示界面参考图设计如下：

```
void LCD_Proc(void)
{
  float temp;

  if(!ucState)                                                 // 液位检测
  {
    LCD_DisplayStringLine(Line1, (u8*)"   Liquid Level    ");

    sprintf((char*)pucStr, "    Height:%03ucm    ", ucHeight);
    LCD_DisplayStringLine(Line4, pucStr);

    temp = (float)uiAdc_Val*3.3/4095;
    sprintf((char*)pucStr, "    VR37:  %4.2fV    ", temp);
    LCD_DisplayStringLine(Line6, pucStr);

    sprintf((char*)pucStr, "    Level: %1u       ", ucLevel);
    LCD_DisplayStringLine(Line8, pucStr);
  }
  else                                                         // 阈值设置
  {
    LCD_DisplayStringLine(Line1, (u8*)"  Threshold Setup   ");

    sprintf((char*)pucStr, "  Threshold 1:%02ucm  ", pucTh[0]);
    if(ucState == 1) LCD_SetBackColor(Red);
    LCD_DisplayStringLine(Line4, pucStr);
    LCD_SetBackColor(Blue);
```



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
    sprintf((char*)pucStr, "  Threshold 2:%02ucm  ", pucTh[1]);
    if(ucState == 2) LCD_SetBackColor(Red);
    LCD_DisplayStringLine(Line6, pucStr);
    LCD_SetBackColor(Blue);

    sprintf((char*)pucStr, "  Threshold 3:%02ucm  ", pucTh[2]);
    if(ucState == 3) LCD_SetBackColor(Red);
    LCD_DisplayStringLine(Line8, pucStr);
    LCD_SetBackColor(Blue);
  }
}
```

### 3）ADC 处理程序设计

ADC 处理程序设计如下：

```
void ADC_Proc(void)
{
  if(!ucState && ucSec != ucSec1)
  {
    ucSec1 = ucSec;

    uiAdc_Val = ADC1_Conv();                                  // 获取转换值
    ucHeight = uiAdc_Val*100/4095;                            // 计算液位高度

    if(ucHeight < pucTh[0])
      ucLevel = 0;
    else if(ucHeight < pucTh[1])
      ucLevel = 1;
    else if(ucHeight < pucTh[2])
      ucLevel = 2;
    else
      ucLevel = 3;

    if(ucLevel != ucLevel1)
    {
      if(ucLevel > ucLevel1)                                  // 液位升高
        printf("A:H%02u+L%1u+U\r\n", ucHeight, ucLevel);
      else                                                    // 液位降低
        printf("A:H%02u+L%1u+D\r\n", ucHeight, ucLevel);

      ucLevel1 = ucLevel;
      ucLd2 = 1;                                              // 等级改变
    }
  }
}
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

4）UART 处理程序设计

UART 处理程序设计如下：

```
void UART_Proc(void)
{
  if(pucRcv[0] == 'C')
    printf("C:H%02u+L%1u\r\n", ucHeight, ucLevel);
  if(pucRcv[0] == 'S')
    printf("S:TL%02u+TM%02u+TH%02u\r\n", pucTh[0], pucTh[1], pucTh[2]);
  if(pucRcv[0] == 'C' || pucRcv[0] == 'S')
    ucLd3 = 1;                                                    // 串口查询
  pucRcv[0] = 0;
}
```

5）LED 处理程序设计

LED 处理用 SysTick 中断处理程序实现，程序设计如下：

```
// SysTick 中断处理程序
void SysTick_Handler(void)
{
  ulTick_ms++;

  if(ulTick_ms%1000 == 0)
  {
    ucSec++;
    ucLed ^= 1;                                                   // LD1 闪烁
  }

  if(ucLd2)                                                       // 等级改变
  {
    if(ulTick_ms%200 == 0)
    {
      if(ucNum--)
        ucLed ^= 2;                                               // LD2 闪烁 5 次
      else
      {
        ucLd2 = 0;
        ucNum = 10;
      }
    }
  }

  if(ucLd3)                                                       // 串口查询
  {
    if(ulTick_ms%200 == 0)
    {
      if(ucNum--)
```



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
            ucLed ^= 4;                                    // LD3 闪烁 5 次
          else
          {
            ucLd3 = 0;
            ucNum = 10;
          }
        }
      }
      LED_Disp(ucLed);
    }
```

6）USART2 中断处理程序设计

USART2 中断处理程序设计如下：

```
// USART2 中断处理程序
void USART2_IRQHandler(void)
{
  pucRcv[0] = USART_ReceiveData(USART2);
}
```

### 6.1.4 系统实现

系统实现的主要步骤如下：

（1）将“206_ADC”文件夹复制并重命名为“301_071”文件夹，在“301_071”文件夹中双击工程文件“STM32.uvproj”打开工程。

（2）修改文件“main.c”的内容。

（3）在 main()后追加 KEY_Proc()、LCD_Proc()、ADC_Proc()和 UART_Proc()的程序代码。

（4）修改 SysTick_Handler()的程序代码。

（5）添加 USART2_IRQHandler()的程序代码。

## 6.2 第七届决赛试题——温、湿度监控设备

温、湿度监控设备通过采集传感器输出电压信号和信号频率得到环境温、湿度数据，并能够根据设定的温、湿度阈值执行相应动作，温、湿度监控设备方框图如图 6.8 所示。

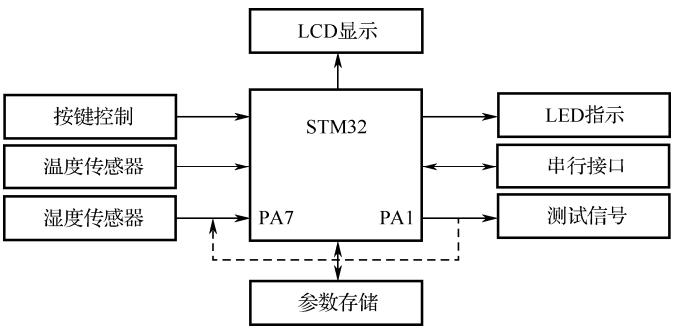



图 6.8 温、湿度监控设备方框图

温、湿度监控设备功能描述如下：

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

1）温度测量功能

通过实训平台上电位器 R37 输出电压模拟温度传感器输出信号，温度测量范围为−20℃～60℃，电压温度关系如下式所示：

$$T = kV_{R37} + b$$

其中，$T$ 代表环境温度，$V_{R37}$ 为电位器输出电压，$k$、$b$ 为常数，输出电压为 0V 时对应温度为−20℃，3.3V 对应温度为 60℃。

2）湿度测量功能

通过实训平台 PA7 引脚检测输入信号频率，相对湿度测量范围为 10%～90%，频率湿度关系如下式所示：

$$H = mF + n$$

其中，$H$ 代表环境湿度，$F$ 为传感器输入设备信号的频率，$m$、$n$ 为常数，1kHz 对应的相对湿度为 10%，10kHz 对应的相对湿度为 90%。

3）测试信号

通过实训平台 PA1 引脚输出频率 1～10kHz 方波，模拟湿度传感器输出信号。

4）参数设置与数据记录功能

可以通过按键设定温、湿度上限和数据采样间隔，温、湿度数据记录间隔可设置为 1～5s，要求至少保存 60 组数据，数据不需要写入 FLASH 或 EEPROM 存储器。

5）按键功能定义

设备上电后通过 LCD 实时显示温、湿度数据和数据记录次数，实时数据界面参考图如图 6.9 所示。

B1 按键：“功能”按键，按下后进入参数设置界面（见图 6.10），再次按下 B1 按键时退出设置界面，保存用户设定的结果到 EEPROM，并返回实时数据界面。

B2 按键：“切换”按键，切换选择 4 个待修改的运行参数，被选中的参数突出显示（如图 6.10 中的“温度上限”）。

| 实时数据 | |
|---|---|
| 当前温度： | −20℃ |
| 当前湿度： | 60% |
| 实时时钟： | 12-50-00 |
| 记录次数： | 20 |

图 6.9　实时数据界面参考图

| 参数设置 | |
|---|---|
| 温度上限： | 40℃ |
| 湿度上限： | 80% |
| 采样间隔： | 1s |
| 测试信号： | 1.5kHz |

图 6.10　参数设置界面参考图

B3 按键：“加”按键，当前选择的参数是温度时，参数加 1℃；选择采样间隔时，参数加 1s；选择参数为湿度时，参数加 5%；选择测试信号时，测试信号频率加 500Hz。

B4 按键：“减”按键，当前选择的参数是温度时，参数减 1℃；选择采样间隔时，参数减 1s，选择参数为湿度时，参数减 5%；选择测试信号时，测试信号频率减 500Hz。

备注：“加”“减”按键应根据当前调整的参数属性，设计合理的边界值。

6）串口功能

设备通过串口完成阈值和数据查询功能，使用 STM32 USART2 完成上述串口功能，波特率设置为 9600。

（1）阈值查询：通过 PC 给设备发送字符“C”，设备返回包含当前温、湿度阈值和当前时间的字符串，格式可自定义。

（2）数据查询：通过 PC 给设备发送字符“T”，设备返回包含当前采集到的所有温、湿度数



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

据的字符串，每条温、湿度数据应包含该条数据的记录时间，格式可自定义。

**7）报警指示功能**

当前温度值超过温度上限时，指示灯 LD1 闪烁报警。

当前湿度值超过湿度上限时，指示灯 LD2 闪烁报警。

每次数据采集时，指示灯 LD3 亮、灭的状态反转。

### 6.2.1 系统设计

通过分析系统功能，可以得到设备详细方框图，如图 6.11 所示。

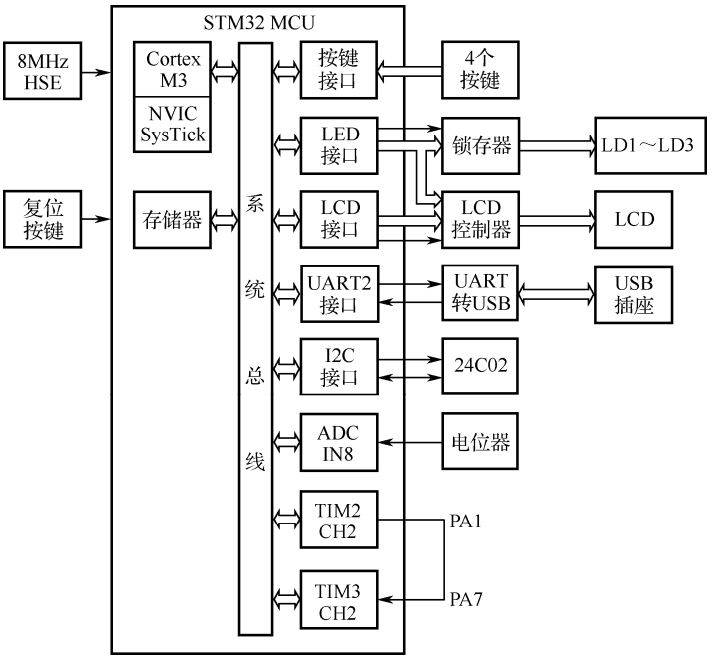



图 6.11　温、湿度监控设备详细方框图

温度测量使用的电位器 R37 连接在 PB0（ADC_IN8），数模转换值与温度的关系为：

$$T = kN + b$$

其中，$T$ 代表环境温度，$N$ 为模数转换值，$k$、$b$ 为常数，$N$=0 对应温度为−20℃，$N$=4095 对应温度为 60℃，代入上式可得数模转换值与温度的关系为：

$$T = 80N/4095 - 20$$

湿度测量通过 PA7 检测输入信号的频率，使用的是 TIM3_CH2 的输入捕捉功能，捕捉值与湿度的关系为：

$$H = m/C + n$$

其中，$H$ 代表环境湿度，$C$ 为捕捉值，$m$、$n$ 为常数，捕捉值的确定与 TIM3 的设置有关。

系统的默认频率是 72MHz，取 TIM3 的预分频值为 72，则计数器的输入频率为 1MHz，1kHz 的捕捉值为 1000，对应湿度为 10%，10kHz 的捕获值为 100，对应湿度为 90%，代入上式可得捕获值与湿度的关系为：

$$H = (80000 / C + 10) / 9$$

测试信号通过 PA1 输出频率 1～10kHz 方波（间隔 500Hz），使用的是 TIM2_CH2 的输出比较功能：系统的默认频率是 72MHz，取 TIM2 的预分频值为 72，则计数器的输入频率为 1MHz，1kHz 的周期值为 1000，10kHz 的周期值为 100，周期值与频率的关系为：

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

$$P = 1000000/F$$

参数存储通过 I2C 接口访问 24C02 实现，串口功能通过 UART2 和 UART 转 USB 实现，按键接口、LED 接口和 LCD 接口均通过 GPIO 实现。

温、湿度监控设备主程序流程图如图 6.12 所示。

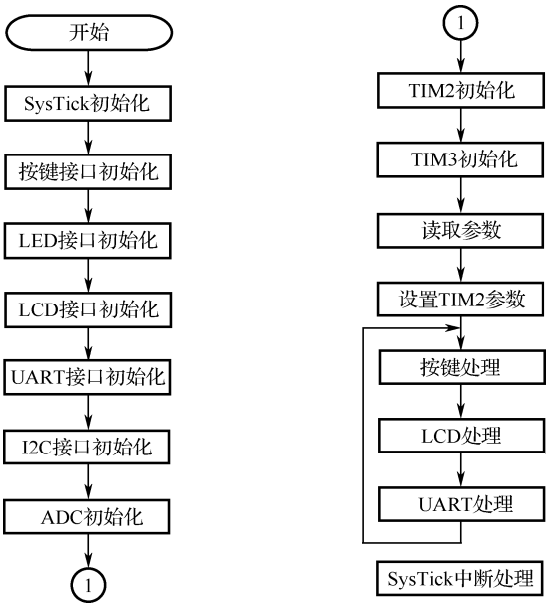



图 6.12　温、湿度监控设备主程序流程图

主程序首先对系统进行初始化，包括 SysTick 初始化、按键接口初始化、LED 接口初始化、LCD 接口初始化、UART 接口初始化、I2C 接口初始化、ADC 初始化、TIM2 初始化和 TIM3 初始化等。系统初始化后从 24C02 中读取参数，并设置 TIM2 参数。

主循环的操作包括按键处理、LCD 处理和 UART 处理等，按键处理中包含保存参数和设置 TIM2 参数，LCD 处理中包含采集温度（ADC 操作）和湿度（TIM3 捕捉），LED 处理在 SysTick 中断处理程序中实现。

主程序设计如下：

```
#include "key.h"
#include "led.h"
#include "lcd.h"
#include "usart.h"
#include "i2c.h"
#include "adc.h"
#include "tim.h"

signed char scTemp, scTemp1;
unsigned char ucHumi, ucHumi1;
unsigned char ucHour, ucMin, ucSec;
unsigned char ucDatano, ucDatanum;
unsigned char ucInte, ucFreq;                          // ucFreq = F/100
unsigned char pucPara[4], pucData[60][5];
unsigned char ucKey_Long, ucLed;
```



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
unsigned char ucState, pucStr[21], pucRcv[1];
unsigned long ulTick_ms, ulSec, ulSec1, ulSec2;

void KEY_Proc(void);
void LCD_Proc(void);
void UART_Proc(void);

int main(void)
{
  SysTick_Config(72000);
  KEY_Init();
  LED_Init();

  STM3210B_LCD_Init();
  LCD_Clear(Blue);
  LCD_SetBackColor(Blue);
  LCD_SetTextColor(White);

  USART2_Init(9600);
  i2c_init();
  ADC1_Init();
  TIM2_OCInit_072();
  TIM3_ICInit_072();

  i2c_read(pucPara, 0, 4);
  scTemp1 = pucPara[0];
  ucHumi1 = pucPara[1];
  ucInte = pucPara[2];
  ucFreq = pucPara[3];
  if(scTemp1 > 60) scTemp1 = 30;
  if((ucHumi1<10) || (ucHumi1>90)) ucHumi1 = 50;
  if((ucInte<1) || (ucInte>5)) ucInte = 1;
  if((ucFreq<10) || (ucFreq>100)) ucFreq = 50;

  TIM_SetAutoreload(TIM2, 10000/ucFreq-1);                    // 周期值
  TIM_SetCompare2(TIM2, 5000/ucFreq);                         // 占空比 1/2

  while(1)
  {
    KEY_Proc();
    LCD_Proc();
    UART_Proc();
  }
}
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

### 6.2.2 初始化程序设计

初始化程序包括 SysTick 初始化、按键接口初始化、LED 接口初始化、LCD 接口初始化、UART 接口初始化、I2C 接口初始化、ADC 初始化、TIM2 初始化和 TIM3 初始化等。

1）SysTick 初始化程序设计

SysTick 初始化用 SysTick_Config()实现。

2）按键接口初始化程序设计

按键接口初始化程序使用“key.c”中的 KEY_Init()实现。

3）LED 接口初始化程序设计

LED 接口初始化程序使用“led.c”中的 LED_Init()实现。

4）LCD 接口初始化程序设计

LCD 接口初始化程序使用“lcd.c”中的 STM3210B_LCD_Init()实现。

5）UART 接口初始化程序设计

UART 接口初始化程序使用“usart.c”中的 USART2_Init()实现。

6）I2C 接口初始化程序设计

I2C 接口初始化程序使用“i2c.c”中的 i2c_init()实现。

7）ADC 初始化程序设计

ADC 初始化程序使用“adc.c”中的 ADC1_Init()实现。

8）TIM2 初始化程序设计

TIM2 初始化程序在“tim.c”中 TIM1_OCInit()的基础上修改实现，内容如下：

```
void TIM2_OCInit_072(void)
{
  GPIO_InitTypeDef GPIO_InitStruct;
  TIM_TimeBaseInitTypeDef TIM_TimeBaseInitStruct;
  TIM_OCInitTypeDef  TIM_OCInitStruct;

  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
  RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2, ENABLE);

  GPIO_InitStruct.GPIO_Pin = GPIO_Pin_1;
  GPIO_InitStruct.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_InitStruct.GPIO_Mode = GPIO_Mode_AF_PP;
  GPIO_Init(GPIOA, &GPIO_InitStruct);

  TIM_TimeBaseInitStruct.TIM_Prescaler = 71;
  TIM_TimeBaseInitStruct.TIM_CounterMode = TIM_CounterMode_Up;
  TIM_TimeBaseInitStruct.TIM_Period = 999;
  TIM_TimeBaseInitStruct.TIM_ClockDivision = TIM_CKD_DIV1;
  TIM_TimeBaseInitStruct.TIM_RepetitionCounter = 0;
  TIM_TimeBaseInit(TIM2, &TIM_TimeBaseInitStruct);

  TIM_OCInitStruct.TIM_OCMode = TIM_OCMode_PWM1;
  TIM_OCInitStruct.TIM_OutputState = TIM_OutputState_Enable;
  TIM_OCInitStruct.TIM_Pulse = 500;
```



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
  TIM_OCInitStruct.TIM_OCPolarity = TIM_OCPolarity_High;
  TIM_OC2Init(TIM2, &TIM_OCInitStruct);

  TIM_Cmd(TIM2, ENABLE);
}
```

### 9）TIM3 初始化程序设计

TIM3 初始化程序在“tim.c”中 TIM2_ICInit()的基础上修改实现，内容如下：

```
void TIM3_ICInit_072(void)
{
  GPIO_InitTypeDef GPIO_InitStruct;
  TIM_TimeBaseInitTypeDef TIM_TimeBaseInitStruct;
  TIM_ICInitTypeDef  TIM_ICInitStruct;

  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
  RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3, ENABLE);

  GPIO_InitStruct.GPIO_Pin = GPIO_Pin_7;
  GPIO_InitStruct.GPIO_Mode = GPIO_Mode_IN_FLOATING;
  GPIO_Init(GPIOA, &GPIO_InitStruct);

  TIM_TimeBaseInitStruct.TIM_Prescaler = 71;
  TIM_TimeBaseInitStruct.TIM_CounterMode = TIM_CounterMode_Up;
  TIM_TimeBaseInitStruct.TIM_Period = 65535;
  TIM_TimeBaseInitStruct.TIM_ClockDivision = TIM_CKD_DIV1;
  TIM_TimeBaseInit(TIM3, &TIM_TimeBaseInitStruct);

  TIM_ICInitStruct.TIM_Channel = TIM_Channel_2;
  TIM_ICInitStruct.TIM_ICPolarity = TIM_ICPolarity_Rising;
  TIM_ICInitStruct.TIM_ICSelection = TIM_ICSelection_DirectTI;
  TIM_ICInitStruct.TIM_ICPrescaler = TIM_ICPSC_DIV1;
  TIM_ICInitStruct.TIM_ICFilter = 0;
  TIM_ICInit(TIM3, &TIM_ICInitStruct);

  TIM_SelectInputTrigger(TIM3, TIM_TS_TI2FP2);
  TIM_SelectSlaveMode(TIM3, TIM_SlaveMode_Reset);
  TIM_Cmd(TIM3, ENABLE);
}

unsigned int TIM3_Cap(void)
{
  if(TIM_GetFlagStatus(TIM3, TIM_FLAG_CC2))
    return TIM_GetCapture2(TIM3)+1;
  else
    return 0;
}
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

### 6.2.3 处理程序设计

处理程序包括按键处理、LCD 处理、UART 处理和 LED 处理等。温、湿度监控设备处理程序流程图如图 6.13 所示。

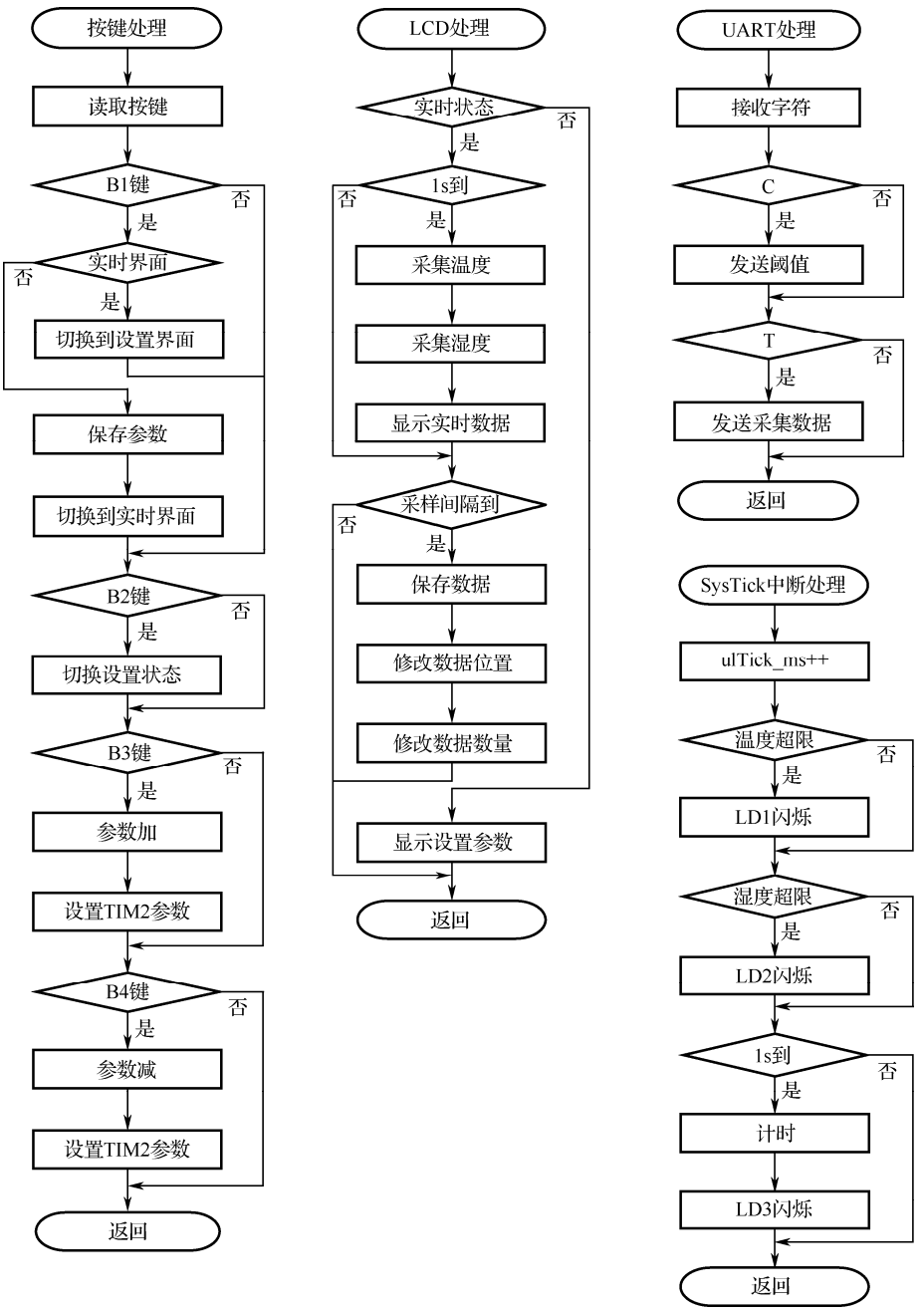



图 6.13　温、湿度监控设备处理程序流程图

#### 1）按键处理程序设计

按键处理程序根据状态图进行设计。温、湿度监控设备工作状态图如图 6.14 所示。

系统共有 5 个状态：1 个实时状态（显示实时数据）和 4 个设置状态（分别设置温度上限、湿度上限、采样间隔和测试频率），实时状态和设置状态间用 B1 键转换，4 个设置状态间用 B2



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

键转换，4 个设置状态均可用 B3/B4 键加减参数。实时状态的 B3/B4 键操作是增加功能，用于实时状态下频率的加减，方便湿度变化的观察。温度变化的观察通过改变电位器 R37 的值实现。

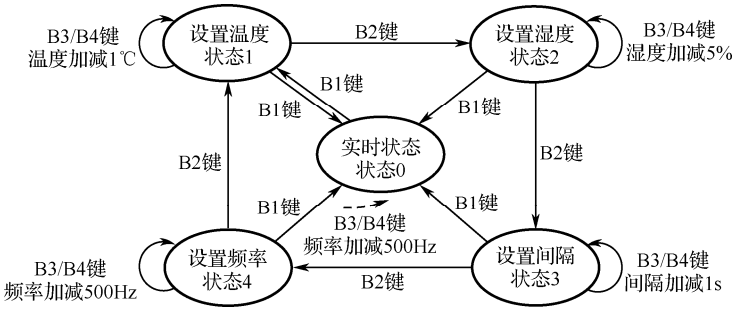



图 6.14　温、湿度监控设备工作状态图

按键处理程序设计如下：

```
void KEY_Proc(void)
{
  unsigned char ucKey_Val;

  ucKey_Val = KEY_Scan();
  if(ucKey_Val != ucKey_Long)
    ucKey_Long = ucKey_Val;
  else
    ucKey_Val = 0;

  switch(ucKey_Val)
  {
    case 1:                                          // B1: 功能键
      if(!ucState)
        ucState = 1;
      else
      {
        pucPara[0] = scTemp1;
        pucPara[1] = ucHumi1;
        pucPara[2] = ucInte;
        pucPara[3] = ucFreq;
        i2c_write(pucPara, 0, 4);                    // 保存参数
        ucState = 0;
        ulSec1 = ulSec;
        ulSec2 = ulSec;
      }break;
    case 2:                                          // B2: 切换键
      if(ucState)
        if(++ucState == 5)
          ucState = 1;
      break;
    case 3:                                          // B3: 加键
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
  if(ucState)                                          // 设置状态
  {
    switch(ucState)
    {
      case 1:
        if(scTemp1 < 60) scTemp1++;                    // 温度上限加 1℃
        break;
      case 2:
        if(ucHumi1 < 90) ucHumi1 += 5;                 // 湿度上限加 5%
        break;
      case 3:
        if(ucInte < 5) ucInte++;                       // 采样间隔加 1s
        break;
      case 4:
        if(ucFreq < 100) ucFreq += 5;                  // 测试频率加 500Hz/100
    }
  }
  else
    if(ucFreq < 100) ucFreq += 5;

  TIM_SetAutoreload(TIM2, 10000/ucFreq-1);             // 周期值
  TIM_SetCompare2(TIM2, 5000/ucFreq);                  // 占空比 1/2
  break;
case 4:                                                // B4: 减键
{
  if(ucState)                                          // 设置状态
  {
    switch(ucState)
    {
      case 1:
        if(scTemp1 > -20) scTemp1--;                   // 温度上限减 1℃
        break;
      case 2:
        if(ucHumi1 > 10) ucHumi1 -= 5;                 // 湿度上限减 5%
        break;
      case 3:
        if(ucInte > 1) ucInte--;                       // 采样间隔减 1s
        break;
      case 4:
        if(ucFreq > 10) ucFreq -= 5;                   // 测试频率减 500Hz/100
    }
  }
  else
    if(ucFreq > 10) ucFreq -= 5;

  TIM_SetAutoreload(TIM2, 10000/ucFreq-1);             // 周期值
```



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
      TIM_SetCompare2(TIM2, 5000/ucFreq);             // 占空比 1/2
    }
  }
}
```

### 2）LCD 处理程序设计

LCD 处理程序根据 LCD 显示界面参考图设计如下：

```
void LCD_Proc(void)
{
  if(!ucState)                                        // 实时数据
  {
    if(ulSec != ulSec1)                               // 1s 到
    {
      ulSec1 = ulSec;

      LCD_DisplayStringLine(Line1, (u8*)"   Real-time Data   ");

      scTemp = (80*ADC1_Conv())/4095-20;              // 采集温度
      sprintf((char*)pucStr, " Temperature:  %3dC ", scTemp);
      LCD_DisplayStringLine(Line3, pucStr);

      ucHumi = (80000/TIM3_Cap()+10)/9;               // 采集湿度
      sprintf((char*)pucStr, " Humidity:    %3d%% ", ucHumi);
      LCD_DisplayStringLine(Line5, pucStr);

      sprintf((char*)pucStr, " Clock:   %02u:%02u:%02u  ",
        ucHour, ucMin, ucSec);
      LCD_DisplayStringLine(Line7, pucStr);

      sprintf((char*)pucStr, " Number:       %2u  ", ucDatanum);
      LCD_DisplayStringLine(Line9, pucStr);
    }

    if((ulSec-ulSec2) >= ucInte)                      // 采样间隔到
    {
      ulSec2 = ulSec;

      pucData[ucDatano][0] = ucHour;                  // 保存时
      pucData[ucDatano][1] = ucMin;                   // 保存分
      pucData[ucDatano][2] = ucSec;                   // 保存秒
      pucData[ucDatano][3] = scTemp;                  // 保存温度
      pucData[ucDatano][4] = ucHumi;                  // 保存湿度

      if(++ucDatano == 60) ucDatano = 0;              // 修改数据位置
      if(ucDatanum != 60) ucDatanum++;                // 修改数据数量
    }
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
  }
  else                                                    // 参数设置
  {
    LCD_DisplayStringLine(Line1, (u8*)"  Parameter Setup   ");

    if(ucState == 1) LCD_SetBackColor(Red);
    sprintf((char*)pucStr, " Temperature:  %3dC ", scTemp1);
    LCD_DisplayStringLine(Line3, pucStr);
    LCD_SetBackColor(Blue);

    if(ucState == 2) LCD_SetBackColor(Red);
    sprintf((char*)pucStr, " Humidity:    %3d%% ", ucHumi1);
    LCD_DisplayStringLine(Line5, pucStr);
    LCD_SetBackColor(Blue);

    if(ucState == 3) LCD_SetBackColor(Red);
    sprintf((char*)pucStr, " Interval:      %1us ", ucInte);
    LCD_DisplayStringLine(Line7, pucStr);
    LCD_SetBackColor(Blue);

    if(ucState == 4) LCD_SetBackColor(Red);
    sprintf((char*)pucStr, " Frequency: %2u.%1ukHz ",
      ucFreq/10, ucFreq%10);
    LCD_DisplayStringLine(Line9, pucStr);
    LCD_SetBackColor(Blue);
  }
}
```

3）串口处理程序设计

串口处理程序设计如下：

```
void UART_Proc(void)
{
  unsigned char i;
  if(pucRcv[0] == 'C')
    printf("C %02u:%02u:%02u %3dC %2u%%\r\n",
      ucHour, ucMin, ucSec, scTemp1, ucHumi1);
  if(pucRcv[0] == 'T')
  {
    for(i=0; i<ucDatanum; i++)                            // 发送存储数据
      printf("T %02u:%02u:%02u %3dC %2u%%\r\n", pucData[i][0],
        pucData[i][1], pucData[i][2], pucData[i][3], pucData[i][4]);
  }
  pucRcv[0] = 0;
}
```



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

#### 4）LED 处理程序设计

LED 处理用 SysTick 中断处理程序实现，程序设计如下：

```
// SysTick 中断处理程序
void SysTick_Handler(void)
{
  ulTick_ms++;

  if(scTemp >= scTemp1)                                    // 温度超限
  {
    if(ulTick_ms%200 == 0)
      ucLed ^= 1;                                          // LD1 闪烁
  }
  else
    ucLed &= 0xfe;

  if(ucHumi >= ucHumi1)                                    // 湿度超限
  {
    if(ulTick_ms%200 == 0)
      ucLed ^= 2;                                          // LD2 闪烁
  }
  else
    ucLed &= 0xfd;

  if(ulTick_ms%1000 == 0)
  {
    ulSec++;
    ucHour = ulSec / 3600;
    ucMin  = ulSec % 3600 / 60;
    ucSec  = ulSec % 3600 % 60;

    ucLed ^= 4;                                            // LD3 闪烁
  }
  LED_Disp(ucLed);
}
```

#### 5）USART2 中断处理程序设计

USART2 中断处理程序设计如下：

```
// USART2 中断处理程序
void USART2_IRQHandler(void)
{
  pucRcv[0] = USART_ReceiveData(USART2);
}
```

### 6.2.4 系统实现

系统实现的主要步骤如下：

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

（1）将“207_TIM”或“301_071”文件夹复制并重命名为“302_072”文件夹，在“032_072”文件夹中双击工程文件“STM32.uvproj”打开工程。

（2）在“tim.h”中添加下列函数声明：

```
void TIM2_OCInit_8272(void);
void TIM3_ICInit_8272(void);
unsigned int TIM3_Cap(void);
```

（3）在“tim.c”中将 TIM1_OCInit()函数代码复制并重命名为 TIM2_OCInit_072()函数代码。

（4）修改 TIM2_OCInit_072()函数代码。

（5）在“tim.c”中将 TIM2_ICInit()函数代码复制并重命名为 TIM3_ICInit()_072 函数代码。

（6）修改 TIM3_ICInit()_072 函数代码。

（7）在“tim.c”中追加 TIM3_Cap()函数代码。

（8）修改文件“main.c”的内容。

（9）在 main()后追加或修改 KEY_Proc()、LCD_Proc()和 UART_Proc()的程序代码。

（10）修改 SysTick_Handler()的程序代码。

（11）添加 USART2_IRQHandler()的程序代码。

**注意：**运行程序时，连接 J3.4（PA1）和 J3.10（PA7）。

## 6.3 第八届预赛试题——模拟升降控制器

模拟升降控制器用来模拟升降机的运行过程。通过按键设定到达目标平台，双路 PWM 输出分别模拟控制上下行电机和开关门电机，LED 流水灯模拟运行方向。模拟升降控制器方框图如图 6.15 所示。

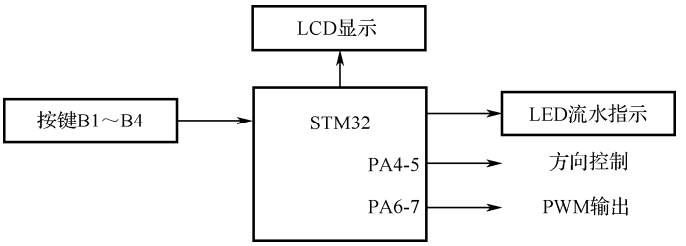



图 6.15　模拟升降控制器方框图

模拟升降控制器功能描述如下：

### 1）LCD 显示

LCD 显示时钟和当前所在平台。当升降机运行到目标平台时，平台数字在 1s 内闪烁 2 次。系统上电后升降机默认位于平台 1，升降机门默认为打开状态。

LCD 显示界面参考图如图 6.16 所示。

图 6.16　LCD 显示界面参考图



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

2）按键功能

4 个按键分别代表 4 个平台。按下后通过点亮 LED 指示要到达的平台（LD1～LD4，LD1 对应平台 1，依次类推），当升降机到达目标平台时，对应 LED 灯熄灭。按下当前平台对应的按键无效。一次可以设定多个目标平台，上下平台都有按键时，运行顺序是先上后下。

在最后一次按键按完 1s 之后，模拟电机开始运行：升降机先关门，运行到下一层后开门；如果还有目标平台，则开门后停留 2s，然后关门，继续运行到下一个目标平台，直到没有目标平台。

当设置多个目标平台时，升降机的升降顺序与按键顺序无关，先按从低到高的顺序走完设定的上行平台，再按从高到低的顺序走完设定的下行平台。

电机运转期间按键无效。

3）PWM 输出

两路 PWM 输出分别控制上下行电机（PA6）和开关门电机（PA7），并分别用一路方向控制信号控制电机的运转方向。PA4 高电平表示升降机上行，反之表示升降机下行。PA5 高电平表示升降机开门，反之表示升降机关门。

上下每层平台的电机运行时间为 6s，升降机开关门时间为 4s。

上下行电机控制信号的频率为 1kHz。升降机上行时，PWM 信号占空比为 80%；升降机下行时，PWM 信号占空比为 60%。

开关门电机控制信号的频率为 2kHz。升降机开门时，PWM 信号占空比为 60%；升降机关门时，PWM 信号占空比为 50%。

4）LED 流水指示

升降机上下行时，4 个 LED（LD5～LD8）组成流水灯用来表示升降机的运行方向。合理选择流水灯的流水方式和时间间隔。

### 6.3.1 系统设计

通过分析系统功能，可以得到模拟升降控制器详细方框图，如图 6.17 所示。

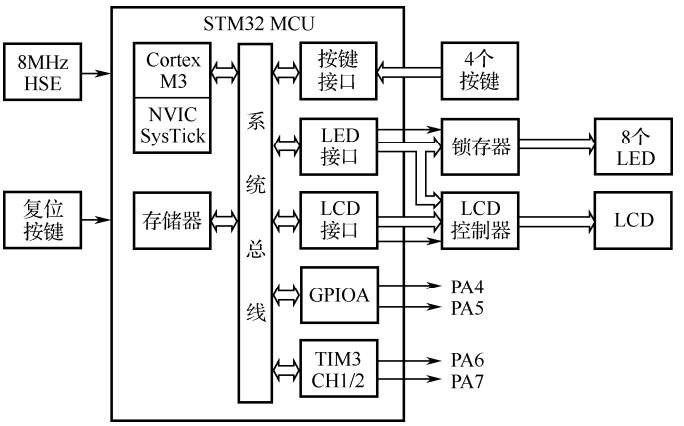



图 6.17　模拟升降控制器详细方框图

4 个按键用于设置 4 个到达平台，设置结果用于 LED 显示和控制升降机运行；8 个 LED 中的 LD1～LD4 用于显示到达平台，LD5～LD8 用于升降机升降时的流水指示；LCD 用于显示当前平台和时间；PA4 用于指示开关门，PA5 用于指示升降，PA6 控制开关门电机，PA7 控制升降电机。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

模拟升降控制器主程序流程图如图 6.18 所示。

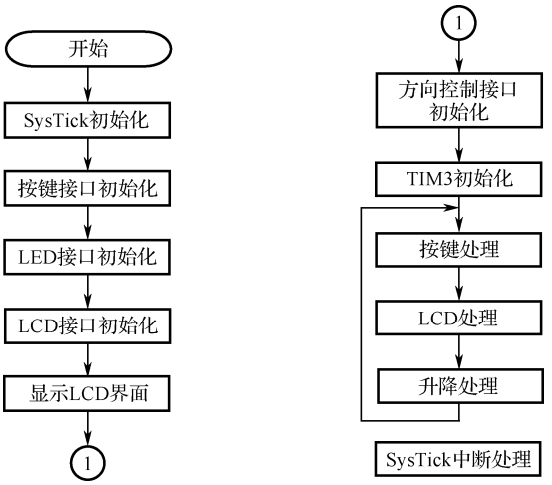



图 6.18　模拟升降控制器主程序流程图

主程序首先对系统进行初始化，包括 SysTick 初始化、按键接口初始化、LED 接口初始化、LCD 接口初始化、方向控制接口初始化和 TIM3 初始化。

主循环的操作包括按键处理、LCD 处理和升降处理等，其中升降处理是系统设计的核心和难点。

初看起来，模拟升降控制器的程序设计比较简单，但进一步分析发现：简单的只是接口和设备的程序设计，核心的升降处理其实并不简单。

主程序设计如下：

```
#include "key.h"
#include "led.h"
#include "lcd.h"
#include "tim.h"
#include "stdio.h"

unsigned char ucPlat = 1, ucSet, ucDir;
unsigned char ucHour, ucMin, ucSec, ucSec1, ucKey_Long;
unsigned char ucState, pucStr[21], ucLed = 1;
unsigned long ulTick_ms, ulTick_ms1, ulSec;

void KEY_Proc(void);
void LCD_Proc(void);
void ELEV_Proc(void);
void Delay_ms(unsigned char ms);

int main(void)
{
  SysTick_Config(72000);
  KEY_Init();
```



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
  LED_Init();

  STM3210B_LCD_Init();
  LCD_Clear(Blue);
  LCD_SetBackColor(Blue);
  LCD_SetTextColor(White);

  LCD_DisplayStringLine(Line2, (u8*)"  Current Platform  ");
  LCD_DisplayChar(Line4, 160, ucPlat+0x30);
  LCD_DisplayStringLine(Line6, (u8*)"    Door Open.    ");

  DIR_Init();
  TIM3_OCInit_081();

  while(1)
  {
    KEY_Proc();
    LCD_Proc();
    ELEV_Proc();
  }
}
```

### 6.3.2 初始化程序设计

初始化程序包括 SysTick 初始化、按键接口初始化、LED 接口初始化、LCD 接口初始化、方向控制接口初始化和 TIM3 初始化。

**1）SysTick 初始化程序设计**

SysTick 初始化用 SysTick_Config()实现。

**2）按键接口初始化程序设计**

按键接口初始化程序使用“key.c”中的 KEY_Init()实现。

**3）LED 接口初始化程序设计**

LED 接口初始化程序使用“led.c”中的 LED_Init()实现。

**4）LCD 接口初始化程序设计**

LCD 接口初始化程序使用“lcd.c”中的 STM3210B_LCD_Init()实现。

**5）方向控制接口初始化程序设计**

方向控制接口初始化程序设计如下：

```
void DIR_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct;

  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
  // PA4-PA5 通用推挽输出
  GPIO_InitStruct.GPIO_Pin = GPIO_Pin_4 | GPIO_Pin_5;
  GPIO_InitStruct.GPIO_Speed = GPIO_Speed_50MHz;
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
  GPIO_InitStruct.GPIO_Mode = GPIO_Mode_Out_PP;
  GPIO_Init(GPIOA, &GPIO_InitStruct);
}
```

6）TIM3 初始化程序设计

TIM3 初始化程序在“tim.c”中 TIM1_OCInit()的基础上修改实现，内容如下：

```
void TIM3_OCInit_081(void)
{
  GPIO_InitTypeDef GPIO_InitStruct;
  TIM_TimeBaseInitTypeDef TIM_TimeBaseInitStruct;
  TIM_OCInitTypeDef  TIM_OCInitStruct;

  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
  RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3, ENABLE);

  GPIO_InitStruct.GPIO_Pin = GPIO_Pin_6 | GPIO_Pin_7;
  GPIO_InitStruct.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_InitStruct.GPIO_Mode = GPIO_Mode_AF_PP;
  GPIO_Init(GPIOA, &GPIO_InitStruct);

  TIM_TimeBaseInitStruct.TIM_Prescaler = 71;
  TIM_TimeBaseInitStruct.TIM_CounterMode = TIM_CounterMode_Up;
  TIM_TimeBaseInitStruct.TIM_Period = 999;
  TIM_TimeBaseInitStruct.TIM_ClockDivision = TIM_CKD_DIV1;
  TIM_TimeBaseInitStruct.TIM_RepetitionCounter = 0;
  TIM_TimeBaseInit(TIM3, &TIM_TimeBaseInitStruct);

  TIM_OCInitStruct.TIM_OCMode = TIM_OCMode_PWM1;
  TIM_OCInitStruct.TIM_OutputState = TIM_OutputState_Disable;
  TIM_OCInitStruct.TIM_Pulse = 500;
  TIM_OCInitStruct.TIM_OCPolarity = TIM_OCPolarity_High;
  TIM_OC1Init(TIM3, &TIM_OCInitStruct);
  TIM_OC2Init(TIM3, &TIM_OCInitStruct);

  TIM_Cmd(TIM3, ENABLE);
}
```

### 6.3.3 处理程序设计

处理程序包括按键处理、LCD 处理和升降处理等。按键处理和升降处理程序流程图如图 6.19 所示。



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

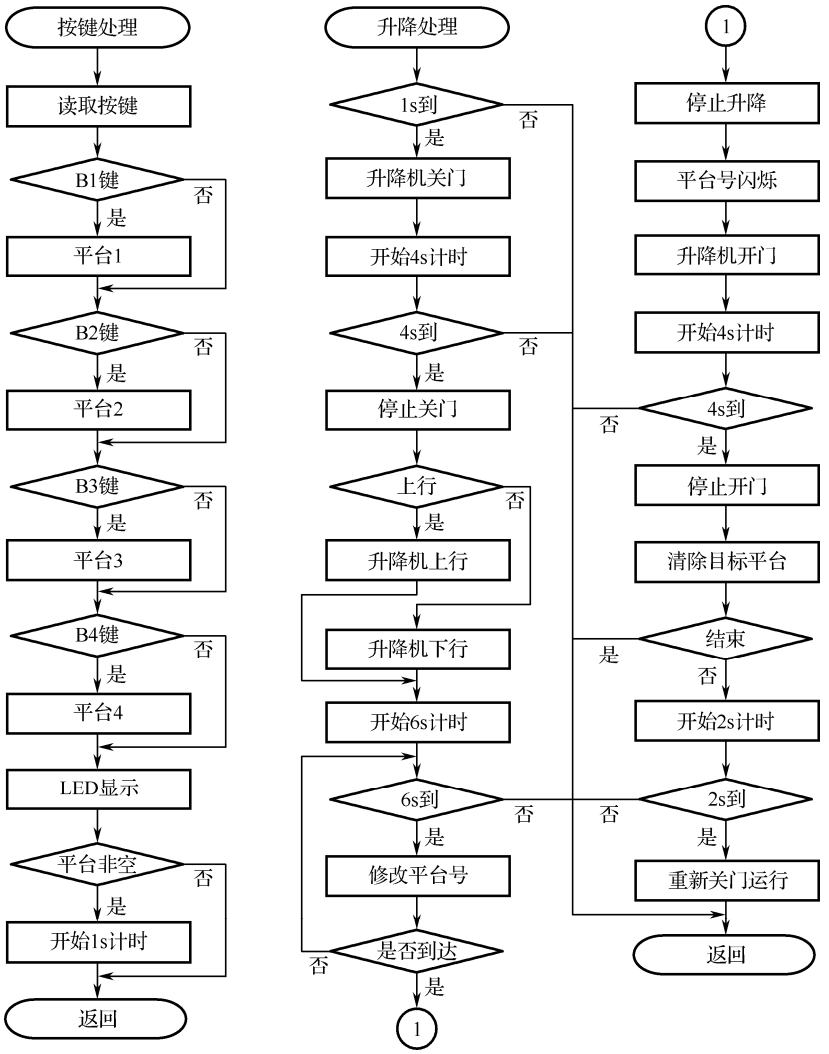



图 6.19　按键处理和升降处理程序流程图

### 1）按键处理程序设计

按键处理程序设计如下：

```
void KEY_Proc(void)
{
  unsigned char ucKey_Val;
  if(!ucState)
  {
    ucKey_Val = KEY_Scan();
    if(ucKey_Val != ucKey_Long)
      ucKey_Long = ucKey_Val;
    else
      ucKey_Val = 0;

    switch(ucKey_Val)
    {
      case 1:                                    // B1: 功能键
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
        if(ucPlat != 1) ucSet |= 1;
        break;
      case 2:                                          // B2: 切换键
        if(ucPlat != 2) ucSet |= 2;
        break;
      case 3:                                          // B3: 加键
        if(ucPlat != 3) ucSet |= 4;
        break;
      case 4:                                          // B4: 减键
        if(ucPlat != 4) ucSet |= 8;
    }
    LED_Disp(ucSet);

    if(ucKey_Val) ulTick_ms1 = ulTick_ms;
  }
}
```

### 2）LCD 处理程序设计

LCD 处理程序设计如下：

```
void LCD_Proc(void)
{
  if(ucSec1 != ucSec)
  {
    ucSec1 = ucSec;
    sprintf((char*)pucStr, "     %02u:%02u:%02u     ",\
      ucHour, ucMin, ucSec);
    LCD_DisplayStringLine(Line8, pucStr);
  }
}
```

### 3）升降处理程序设计

升降处理程序根据状态图进行设计。升降机工作状态图如图 6.20 所示。

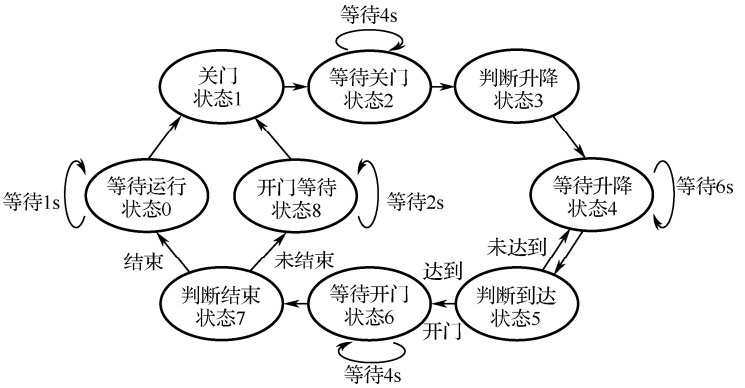



图 6.20　升降机工作状态图

系统共有 9 个状态：5 个等待状态、1 个关门状态和 3 个判断状态。判断升降状态根据方向控制升降机升降，等待升降期间，LD5～LD8 流水显示升降机的运行方向。到达平台时停止升降，



电子工业出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

平台号闪烁，升降机开门，未到达平台时继续升降。升降机开门后清除目标平台，并判断是否结束，未结束时开门等待 2s 后重新关门运行，结束时退出处理程序。

升降处理程序设计如下：

```
void ELEV_Proc(void)
{
  if(ucSet)
  {
    switch(ucState)
    {
      case 0:                                              // 等待运行
        if(ulTick_ms-ulTick_ms1 >= 1000)
          ucState = 1;
        else
          break;
      case 1:                                              // 关门
        LCD_DisplayStringLine(Line6, (u8*)"    Door Close...  ");
        GPIO_ResetBits(GPIOA, GPIO_Pin_5);

        TIM_SetAutoreload(TIM3, 499);                      // PA7 频率 2kHz
        TIM_SetCompare2(TIM3, 250);                        // 占空比 50%
        TIM_CCxCmd(TIM3, TIM_Channel_2, ENABLE);

        ulTick_ms1 = ulTick_ms;
        ucState = 2;
      case 2:                                              // 等待关门
        if(ulTick_ms-ulTick_ms1 >= 4000)
        {
          TIM_CCxCmd(TIM3, TIM_Channel_2, DISABLE);
          ucState = 3;
        }
        else
          break;
      case 3:                                              // 升降
        if(ucSet > (1<<(ucPlat-1)))
        {
          ucDir = 1;
          LCD_DisplayStringLine(Line6, (u8*)"  Elevator Up...    ");
          GPIO_SetBits(GPIOA, GPIO_Pin_4);

          TIM_SetAutoreload(TIM3, 999);                    // PA6 频率 1kHz
          TIM_SetCompare1(TIM3, 800);                      // 占空比 80%
        }
        else
        {
          ucDir = 2;
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
    LCD_DisplayStringLine(Line6, (u8*)"  Elevator Down...   ");
    GPIO_ResetBits(GPIOA, GPIO_Pin_4);

    TIM_SetAutoreload(TIM3, 999);                    // PA6 频率 1kHz
    TIM_SetCompare1(TIM3, 600);                      // 占空比 60%
  }
  TIM_CCxCmd(TIM3, TIM_Channel_1, ENABLE);
  ulTick_ms1 = ulTick_ms;
  ucState = 4;
case 4:                                              // 等待升降
  if(ulTick_ms-ulTick_ms1 >= 6000)
  {
    if(ucDir == 1) ucPlat++;
    if(ucDir == 2) ucPlat--;
    LCD_DisplayChar(Line4, 160, ucPlat+0x30);
    ucState = 5;
  }
  else
  {
    if(ucDir == 1)
    {
      ucLed <<= 1;                                   // LED 左环移
      if(ucLed == 0x10) ucLed = 1;
    }
    if(ucDir == 2)
    {
      ucLed >>= 1;                                   // LED 右环移
      if(ucLed == 0) ucLed = 8;
    }
    LED_Disp(ucSet + (ucLed<<4));
    Delay_ms(250);
    break;
  }
case 5:                                              // 判断是否达到
  if(ucSet & (1<<(ucPlat-1)))
  {
    TIM_CCxCmd(TIM3, TIM_Channel_1, DISABLE);

    LCD_DisplayChar(Line4, 160, ' ');
    Delay_ms(250);
    LCD_DisplayChar(Line4, 160, ucPlat+0x30);
    Delay_ms(250);
    LCD_DisplayChar(Line4, 160, ' ');
    Delay_ms(250);
    LCD_DisplayChar(Line4, 160, ucPlat+0x30);
    Delay_ms(250);
```



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
    LCD_DisplayStringLine(Line6, (u8*)"    Door Open...   ");
    GPIO_SetBits(GPIOA, GPIO_Pin_5);

    TIM_SetAutoreload(TIM3, 499);                // PA7 频率 2kHz
    TIM_SetCompare2(TIM3, 300);                  // 占空比 60%
    TIM_CCxCmd(TIM3, TIM_Channel_2, ENABLE);

    ulTick_ms1 = ulTick_ms;
    ucState = 6;
  }
  else
  {
    ulTick_ms1 = ulTick_ms;
    ucState = 4;
    break;
  }
case 6:                                          // 等待开门
  if(ulTick_ms-ulTick_ms1 >= 4000)
  {
    ucDir = 0;

    LCD_DisplayStringLine(Line6, (u8*)"    Door Open.     ");
    TIM_CCxCmd(TIM3, TIM_Channel_2, DISABLE);

    ucSet &= ~(1<<(ucPlat-1));
    LED_Disp(ucSet);

    ucState = 7;
  }
  else
    break;
case 7:                                          // 判断是否结束
  if(ucSet)
  {
    ulTick_ms1 = ulTick_ms;
    ucState = 8;
  }
  else
  {
    ucState = 0;
    break;
  }
case 8:                                          // 开门等待
  if(ulTick_ms-ulTick_ms1 >= 2000)
    ucState = 1;
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
    }
  }
}

void Delay_ms(unsigned char ms)
{
  unsigned long ms1 = ulTick_ms;

  while(ulTick_ms-ms1 < ms);
}

void SysTick_Handler(void)
{
  ulTick_ms++;
  if(ulTick_ms%1000 == 0)
  {
    ulSec++;
    ucHour = ulSec / 3600;
    ucMin  = ulSec % 3600 / 60;
    ucSec  = ulSec % 3600 % 60;
  }
}
```

### 6.3.4 系统实现

系统实现的主要步骤如下：

（1）将“207_TIM”文件夹复制并重命名为“303_081”文件夹，在“303_081”文件夹中双击工程文件“STM32.uvproj”打开工程。

（2）在“led.h”中添加下列函数声明：

```
void DIR_Init(void);
```

（3）在“led.c”中添加 DIR_Init()函数代码。

（4）在“tim.h”中添加下列函数声明：

```
void TIM3_OCInit_081(void);
```

（5）在“tim.c”中将 TIM1_OCInit()函数代码复制并重命名为 TIM3_OCInit_081()函数代码。

（6）修改 TIM3_OCInit_081()程序代码。

（7）修改文件“main.c”的内容。

（8）在 main()后追加 KEY_Proc()、LCD_Proc()、ELEV_Proc()和 Delay_ms()的程序代码。

（9）修改 SysTick_Handler()的程序代码。

### 6.3.5 客观题

（1）逻辑表达式 $F = A\overline{B} + BC + ABCD + \overline{B}$ 的最简式为（　　）。

A．$C + D$　　B．$C$

C．$\overline{B}+C$　　D．$\overline{A}+C$



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

（2）能够实现线与功能的门电路是（　　）。

A．与非门　　B．或非门

C．异或门　　D．OC 门

（3）STM32F103RBT6 微控制器的内核是（　　）。

A．Cortex-M0　　B．Cortex-M7

C．Cortex-M3　　D．Cortex-M4

（4）STM32F103RBT6 微控制器不支持的数据类型是（　　）。

A．字　　B．双字

C．字节　　D．半字

（5）采用 RS-232 接口实现双机通信，至少需要多少根信号线？（　　）

A．2　　B．3

C．4　　D．5

（6）要以较少的信号线下载并调试 STM32 程序，应该采用哪种方式？（　　）

A．RS-232　　B．JTAG

C．SWJ　　D．SPI

（7）能够以菊花链方式连接的接口是（　　）。

A．RS-232　　B．SPI

C．I2C　　D．JTAG

（8）用 STM32 开发 USB 外设，应该采用哪种时钟源？（　　）

A．HSE　　B．HSI

C．LSE　　D．LSI

（9）下列关于 DMA 描述不正确的选项是（　　）。

A．DMA 可以和 MCU 并行工作

B．DMA 数据传输结束后产生中断

C．DMA 控制器需要经过 MCU 访问内存

D．开始 DMA 传输前需要经过 MCU 初始化

（10）运放电路如图 6.21 所示，输入电压 $U_{I1}$ 和 $U_{I2}$ 分别为 0.3V 和 0.2V，输出电压 $U_O$ 的值是（　　）V。

A．−4.5　　B．−4.0

C．−3.5　　D．−3.0

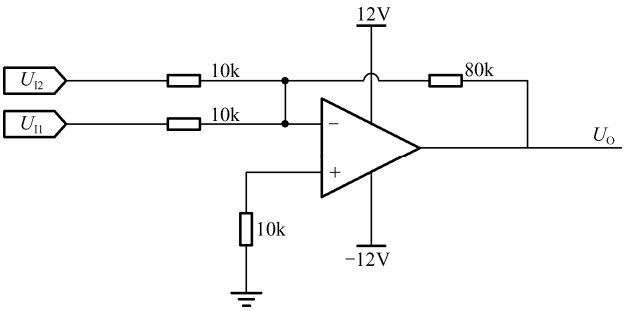



图 6.21　运放电路

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

参考答案与评分标准

（1）C（3分）　（2）D（3分）　（3）C（3分）　（4）B（3分）　（5）B（3分）

（6）C（3分）　（7）BD（3分）　（8）A（3分）　（9）C（3分）　（10）B（3分）

## 6.4　第八届决赛试题——频率控制器

频率控制器方框图如图 6.22 所示。

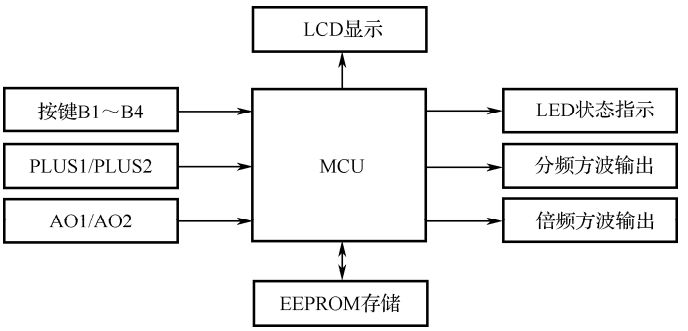



图 6.22　频率控制器方框图

其中分频方波输出（PA6）是 PLUS1（PA1）的 1～4 分频，倍频方波输出（PA7）是 PLUS2（PA2）的 1～4 倍频。

频率控制器功能描述如下：

1）LCD 显示

初始状态下，LCD 显示测量界面。LCD 测量界面参考图如图 6.23 所示。

图 6.23　LCD 测量界面参考图

其中，PLUS1 为脉冲信号 PULS1 的频率，PLUS2 为脉冲信号 PULS2 的频率。频率测量范围应覆盖扩展板上 PULS1 和 PULS2 信号的输出范围。AO1 为电位器 RP5 的输出电压值，AO2 为电位器 RP6 的输出电压值。电压值保留小数点后 2 位有效数字。1 为测量界面数字编号。

LCD 配置界面参考图如图 6.24 所示。

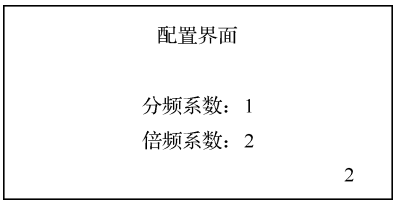



图 6.24　LCD 配置界面参考图

其中，分频系数和倍频系数的取值范围为 1～4。2 为配置界面数字编号。

2）按键功能

按键 B1 为设置键：按下时进入配置界面，再次按下时保存当前设置，切换回测量界面。



电子工业出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

按键 B2 为选择键：按下可切换待配置的参数，被选择的参数项需高亮显示。

按键 B3 为加键：按下时当前选择的参数加 1。

按键 B4 为减键：按下时当前选择的参数减 1。

按键 B3 和 B4 仅在配置界面有效，加减操作应做合理的数据边界保护。

参数配置时分频和倍频停止输出，保持低电平。

**3）参数存储**

分频系数和倍频系数存储在 EEPROM 中，设备上电后重装参数。

**4）LED 显示**

LD1 为参数设置指示：进入配置界面时 LD1 点亮，退出后 LD1 熄灭。

LD8 为电压比较指示：$V_{AO1} > V_{AO2}$ 时 LD8 点亮，反之 LD8 熄灭。

**5）资源扩展板跳线配置**

资源扩展板跳线配置如图 6.25 所示。

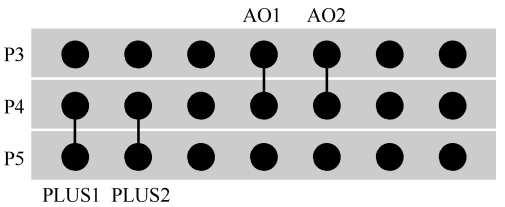



图 6.25　资源扩展板跳线配置

## 6.4.1　系统设计

通过分析系统功能，可以得到频率控制器详细方框图，如图 6.26 所示。

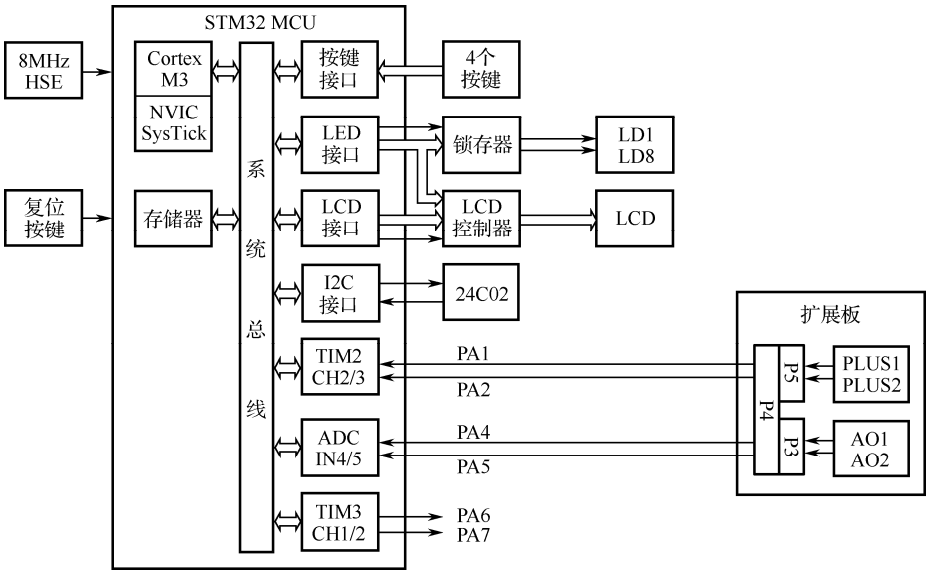



图 6.26　频率控制器详细方框图

扩展板上的 PLUS1 和 PLUS2 分别通过 PA1 和 PA2 与 TIM2 CH2 和 CH3 连接，AO1 和 AO2 分别通过 PA4 和 PA5 与 ADC IN4 和 IN5 连接，分频方波输出和倍频方波输出分别由 TIM3 CH1 和 CH2 产生，并分别通过 PA6 和 PA7 输出。

因为 PLUS1 和 PLUS2 不同步且频率不同，所以只能分别进行捕捉，具体方法是分别将两次

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

相邻的捕捉值相减得到周期值。TIM2 使用的是向上计数模式，当计数值溢出时必须对捕捉值做特殊处理。

同样，分频方波输出和倍频方波输出频率也不同，也只能分别进行比较，具体方法是当比较匹配时输出翻转，并分别将比较值增加半个周期值，当比较值溢出时必须对比较值特殊处理。

PLUS1 和 PLUS2 的频率范围是 720Hz～28.8kHz，分频值是 100000～2500，根据预分频值计算公式

$$\text{int}(\text{分频值} / 65536) < \text{预分频值} \leqslant \min(\text{分频值}/2, 65536)$$

可以计算出 700Hz 的预分频取值范围是 2～50000，28.8kHz 的预分频取值范围是 1～1250，因此 TIM2 的预分频取值范围是 2～1250，可以取值 72。

4 分频后的频率范围是 180Hz～7.2kHz，分频值是 400000～10000，180Hz 的预分频取值范围是 7～65536，7.2kHz 的预分频取值范围是 1～5000，因此 TIM2 CH1 的预分频取值范围是 7～5000。

4 倍频后的频率范围是 2.88～115.2kHz，分频值是 25000～625，2.88kHz 的预分频取值范围是 1～12500，115.2kHz 的预分频取值范围是 1～312，因此 TIM2 CH2 的预分频取值范围是 1～312。

综合考虑 TIM2 CH1 和 CH2 的预分频取值范围，可以得到 TIM2 的预分频取值范围是 7～312，也可以取值 72。

频率控制器主程序流程图如图 6.27 所示。

主程序首先对系统进行初始化，包括 SysTick 初始化、按键接口初始化、LED 接口初始化、LCD 接口初始化、I2C 接口初始化、ADC 初始化、TIM2 和 TIM3 初始化等。系统初始化后从 AT24C02 中读取分频系数和倍频系数。

主循环的操作包括按键处理、LED 处理、LCD 处理、TIM2 处理和 TIM3 处理等，按键处理中包含保存参数，LCD 处理中包含读取 ADC 转换值。

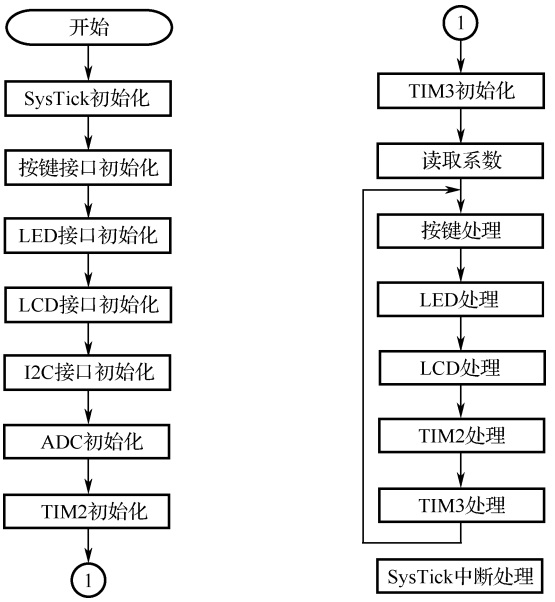



图 6.27　频率控制器主程序流程图

主程序设计如下：

```
#include "key.h"
#include "led.h"
#include "lcd.h"
```



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
#include "stdio.h"
#include "i2c.h"
#include "adc.h"
#include "tim.h"
#define TIM_CK 1000000

unsigned int  uiAo1, uiAo2, uiId1, uiId2;
unsigned long ulIc1, ulIc2, ulOc1, ulOc2;
unsigned char ucDiv = 1, ucMul = 1;
unsigned char ucState, pucStr[21], ucLed, ucKey_Long;
unsigned long ulTick_ms, ulSec, ulSec1;

void KEY_Proc(void);
void LED_Proc(void);
void LCD_Proc(void);
void TIM2_Proc(void);
void TIM3_Proc(void);

int main(void)
{
  SysTick_Config(72000);
  KEY_Init();
  LED_Init();

  STM3210B_LCD_Init();
  LCD_Clear(Blue);
  LCD_SetBackColor(Blue);
  LCD_SetTextColor(White);

  i2c_init();
  ADC1_Init_AKEY();
  TIM2_ICInit_082();
  TIM3_OCInit_082();

  i2c_read(pucStr, 0, 2);
  ucDiv = pucStr[0];
  ucMul = pucStr[1];
  if((ucDiv<1) || (ucDiv>5)) ucDiv = 1;
  if((ucMul<1) || (ucMul>5)) ucMul = 1;

  while(1)
  {
    KEY_Proc();
    LED_Proc();
    LCD_Proc();
    TIM2_Proc();
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
        TIM3_Proc();
      }
    }
```

### 6.4.2 初始化程序设计

初始化程序包括 SysTick 初始化、按键接口初始化、LED 接口初始化、LCD 接口初始化、I2C 接口初始化、ADC 初始化、TIM2 初始化和 TIM3 初始化等。

1）SysTick 初始化程序设计

SysTick 初始化用 SysTick_Config()实现。

2）按键接口初始化程序设计

按键接口初始化程序使用“key.c”中的 KEY_Init()实现。

3）LED 接口初始化程序设计

LED 接口初始化程序使用“led.c”中的 LED_Init()实现。

4）LCD 接口初始化程序设计

LCD 接口初始化程序使用“lcd.c”中的 STM3210B_LCD_Init()实现。

5）I2C 接口初始化程序设计

I2C 接口初始化程序使用“i2c.c”中的 i2c_init()实现。

6）ADC 初始化程序设计

ADC 初始化程序在“adc.c”中 ADC1_Init_AKEY()的基础上修改实现，内容如下：

```
void ADC1_Init_AKEY(void)
{
  GPIO_InitTypeDef GPIO_InitStruct;
  ADC_InitTypeDef ADC_InitStruct;

  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1, ENABLE);

  GPIO_InitStruct.GPIO_Pin = GPIO_Pin_4 | GPIO_Pin_5;
  GPIO_InitStruct.GPIO_Mode = GPIO_Mode_AIN;
  GPIO_Init(GPIOA, &GPIO_InitStruct);

  ADC_InitStruct.ADC_Mode = ADC_Mode_Independent;
  ADC_InitStruct.ADC_ScanConvMode = DISABLE;
  ADC_InitStruct.ADC_ContinuousConvMode = DISABLE;
  ADC_InitStruct.ADC_ExternalTrigConv = ADC_ExternalTrigConv_None;
  ADC_InitStruct.ADC_DataAlign = ADC_DataAlign_Right;
  ADC_InitStruct.ADC_NbrOfChannel = 1;
  ADC_Init(ADC1, &ADC_InitStruct);

  ADC_RegularChannelConfig(ADC1, ADC_Channel_5, 1,
    ADC_SampleTime_1Cycles5);
  ADC_InjectedChannelConfig(ADC1, ADC_Channel_4, 1,
    ADC_SampleTime_1Cycles5);
```



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
  ADC_AutoInjectedConvCmd(ADC1, ENABLE);

  ADC_Cmd(ADC1, ENABLE);
  ADC_StartCalibration(ADC1);
  while(ADC_GetCalibrationStatus(ADC1));
}
```

### 7）TIM2 初始化程序设计

TIM2 初始化程序在“tim.c”中 TIM2_ICInit()的基础上修改实现，内容如下：

```
void TIM2_ICInit_082(void)
{
  GPIO_InitTypeDef GPIO_InitStruct;
  TIM_TimeBaseInitTypeDef TIM_TimeBaseInitStruct;
  TIM_ICInitTypeDef  TIM_ICInitStruct;
//NVIC_InitTypeDef NVIC_InitStruct;

  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
  RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2, ENABLE);

  GPIO_InitStruct.GPIO_Pin = GPIO_Pin_1 | GPIO_Pin_2;
  GPIO_InitStruct.GPIO_Mode = GPIO_Mode_IN_FLOATING;
  GPIO_Init(GPIOA, &GPIO_InitStruct);

  TIM_TimeBaseInitStruct.TIM_Prescaler = 71;
  TIM_TimeBaseInitStruct.TIM_CounterMode = TIM_CounterMode_Up;
  TIM_TimeBaseInitStruct.TIM_Period = 65535;
  TIM_TimeBaseInitStruct.TIM_ClockDivision = TIM_CKD_DIV1;
  TIM_TimeBaseInit(TIM2, &TIM_TimeBaseInitStruct);

  TIM_ICInitStruct.TIM_Channel = TIM_Channel_2;
  TIM_ICInitStruct.TIM_ICPolarity = TIM_ICPolarity_Rising;
  TIM_ICInitStruct.TIM_ICSelection = TIM_ICSelection_DirectTI;
  TIM_ICInitStruct.TIM_ICPrescaler = TIM_ICPSC_DIV1;
  TIM_ICInitStruct.TIM_ICFilter = 0;
  TIM_ICInit(TIM2, &TIM_ICInitStruct);

  TIM_ICInitStruct.TIM_Channel = TIM_Channel_3;
  TIM_ICInit(TIM2, &TIM_ICInitStruct);

  TIM_Cmd(TIM2, ENABLE);
/*中断初始化
  TIM_ITConfig(TIM2, TIM_IT_CC2 | TIM_IT_CC3, ENABLE);
  NVIC_InitStruct.NVIC_IRQChannel = TIM2_IRQn;
  NVIC_InitStruct.NVIC_IRQChannelPreemptionPriority = 0;
  NVIC_InitStruct.NVIC_IRQChannelSubPriority = 0;
  NVIC_InitStruct.NVIC_IRQChannelCmd = ENABLE;
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
  NVIC_Init(&NVIC_InitStruct);*/
}
```

8）TIM3 初始化程序设计

TIM3 初始化程序在“tim.c”中 TIM1_OCInit()的基础上修改实现，内容如下：

```
void TIM3_OCInit_082(void)
{
  GPIO_InitTypeDef GPIO_InitStruct;
  TIM_TimeBaseInitTypeDef TIM_TimeBaseInitStruct;
  TIM_OCInitTypeDef  TIM_OCInitStruct;
//NVIC_InitTypeDef NVIC_InitStruct;

  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
  RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3, ENABLE);

  GPIO_InitStruct.GPIO_Pin = GPIO_Pin_6 | GPIO_Pin_7;
  GPIO_InitStruct.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_InitStruct.GPIO_Mode = GPIO_Mode_AF_PP;
  GPIO_Init(GPIOA, &GPIO_InitStruct);

  TIM_TimeBaseInitStruct.TIM_Prescaler = 71;
  TIM_TimeBaseInitStruct.TIM_CounterMode = TIM_CounterMode_Up;
  TIM_TimeBaseInitStruct.TIM_Period = 65535;
  TIM_TimeBaseInitStruct.TIM_ClockDivision = TIM_CKD_DIV1;
  TIM_TimeBaseInitStruct.TIM_RepetitionCounter = 0;
  TIM_TimeBaseInit(TIM3, &TIM_TimeBaseInitStruct);

  TIM_OCInitStruct.TIM_OCMode = TIM_OCMode_Toggle;
  TIM_OCInitStruct.TIM_OutputState = TIM_OutputState_Enable;
  TIM_OCInitStruct.TIM_Pulse = 250;
  TIM_OCInitStruct.TIM_OCPolarity = TIM_OCPolarity_High;
  TIM_OC1Init(TIM3, &TIM_OCInitStruct);
  TIM_OC2Init(TIM3, &TIM_OCInitStruct);

  TIM_Cmd(TIM3, ENABLE);
/*中断初始化
  TIM_ITConfig(TIM3, TIM_IT_CC1 | TIM_IT_CC2, ENABLE);
  NVIC_InitStruct.NVIC_IRQChannel = TIM3_IRQn;
  NVIC_InitStruct.NVIC_IRQChannelPreemptionPriority = 0;
  NVIC_InitStruct.NVIC_IRQChannelSubPriority = 0;
  NVIC_InitStruct.NVIC_IRQChannelCmd = ENABLE;
  NVIC_Init(&NVIC_InitStruct);*/
}
```



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

### 6.4.3 处理程序设计

处理程序包括按键处理、LED 处理、LCD 处理、TIM2 处理和 TIM3 处理等，按键处理、TIM2 处理和 TIM3 处理程序流程图如图 6.28 所示。

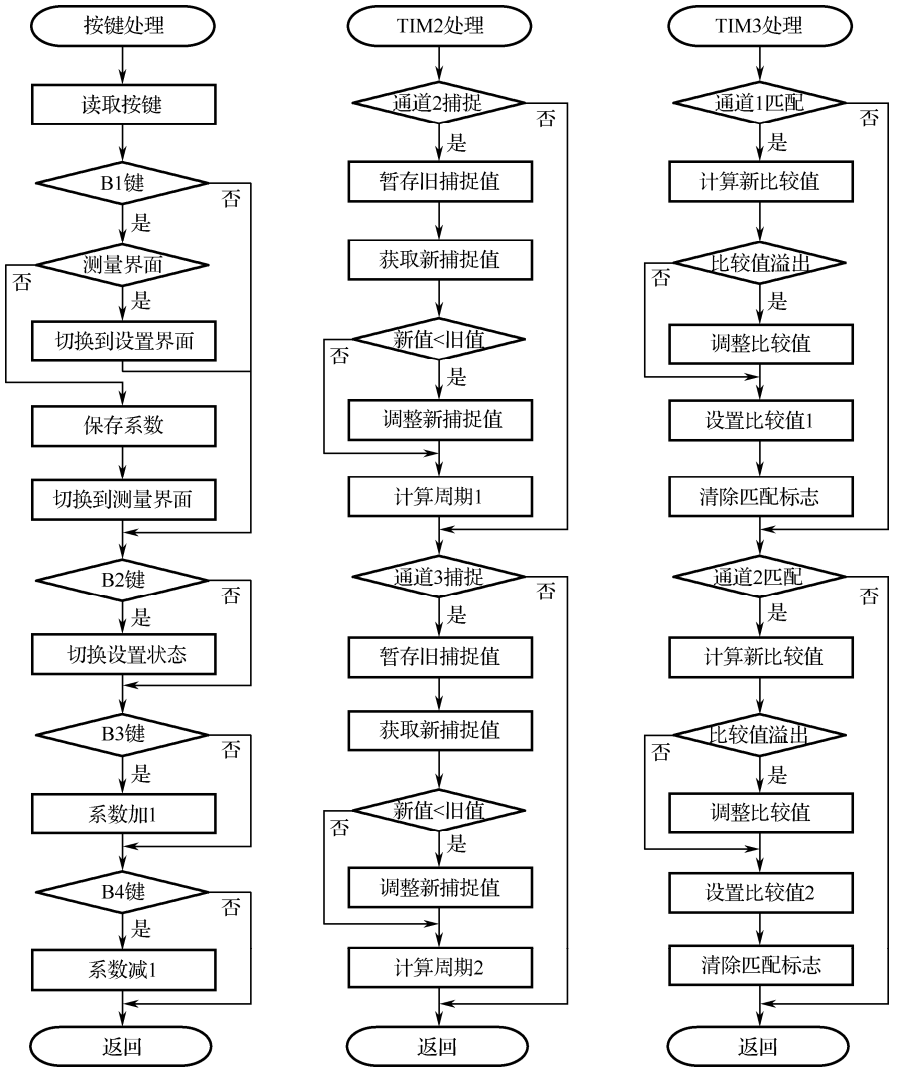



图 6.28　频率控制器处理程序流程图

#### 1）按键处理程序设计

按键处理程序设计如下：

```
void KEY_Proc(void)
{
  unsigned char ucKey_Val;
  ucKey_Val = KEY_Scan();
  if(ucKey_Val != ucKey_Long)
    ucKey_Long = ucKey_Val;
  else
    ucKey_Val = 0;
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
  switch(ucKey_Val)
  {
    case 1:                                                    // B1: 设置键
      if(!ucState)
        ucState = 1;
      else
      {
        pucStr[0] = ucDiv;
        pucStr[1] = ucMul;
        i2c_write(pucStr, 0, 2);

        ucState = 0;
      }break;
    case 2:                                                    // B2: 切换键
      if(ucState)
        if(++ucState == 3)
          ucState = 1;
      break;
    case 3:                                                    // B3: 系数加
      if(ucState == 1)
        if(ucDiv < 4) ucDiv++;
      if(ucState == 2)
        if(ucMul < 4) ucMul++;
      break;
    case 4:                                                    // B4: 系数减
      if(ucState == 1)
        if(ucDiv > 1) ucDiv--;
      if(ucState == 2)
        if(ucMul > 1) ucMul--;
  }
}
```

### 2）LED 处理程序设计

LED 处理程序设计如下：

```
void LED_Proc(void)
{
  if(ucState)
    ucLed |= 1;
  else
    ucLed &= 0xfe;

  if(uiAo1 > uiAo2)
    ucLed |= 0x80;
  else
    ucLed &= 0x7f;
```



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
  LED_Disp(ucLed);
}
```

### 3）LCD 处理程序设计

LCD 处理程序根据 LCD 显示界面参考图设计如下：

```
void LCD_Proc(void)
{
  if(!ucState)                                              // 测量界面
  {
    if(ulSec1 != ulSec)
    {
      ulSec1 = ulSec;

      LCD_DisplayStringLine(Line1, (u8*)"     Measure      ");

      sprintf((char*)pucStr, " %5dHz /%1d %5dHz ",
        TIM_CK/uiId1, ucDiv, TIM_CK/uiId1/ucDiv);
      LCD_DisplayStringLine(Line3, pucStr);

      sprintf((char*)pucStr, " %5dHz *%1d %5dHz ",
        TIM_CK/uiId2, ucMul, TIM_CK/uiId2*ucMul);
      LCD_DisplayStringLine(Line5, pucStr);

      uiAo1 = ADC1_InjeConv();
      sprintf((char*)pucStr, "    AO1:%4.2fV     ", uiAo1*3.3/4095);
      LCD_DisplayStringLine(Line7, pucStr);

      uiAo2 = ADC1_Conv();
      sprintf((char*)pucStr, "    AO2:%4.2fV    1 ", uiAo2*3.3/4095);
      LCD_DisplayStringLine(Line9, pucStr);
    }
  }
  else                                                      // 设置界面
  {
    LCD_DisplayStringLine(Line1, (u8*)"      Setup       ");

    sprintf((char*)pucStr, "     DIV:  %1d      ", ucDiv);
    if(ucState == 1) LCD_SetBackColor(Red);
    LCD_DisplayStringLine(Line3, pucStr);
    LCD_SetBackColor(Blue);

    sprintf((char*)pucStr, "     MUL:  %1d      ", ucMul);
    if(ucState == 2) LCD_SetBackColor(Red);
    LCD_DisplayStringLine(Line5, pucStr);
    LCD_SetBackColor(Blue);
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
    LCD_DisplayStringLine(Line7, (u8*)"                  ");
    LCD_DisplayStringLine(Line9, (u8*)"               2 ");
  }
}
```

4）TIM2 处理程序设计

TIM2 处理程序设计如下：

```
void TIM2_Proc(void)
//void TIM2_IRQHandler(void)
{
  unsigned long ic;

  if(TIM_GetFlagStatus(TIM2, TIM_FLAG_CC2))
  {
    ic = ulIc1;
    ulIc1 = TIM_GetCapture2(TIM2);
    if(ulIc1 > ic)
      uiId1 = ulIc1 - ic;
    else
      uiId1 = 65536 + ulIc1 - ic;
  }
  if(TIM_GetFlagStatus(TIM2, TIM_FLAG_CC3))
  {
    ic = ulIc2;
    ulIc2 = TIM_GetCapture3(TIM2);
    if(ulIc2 > ic)
      uiId2 = ulIc2 - ic;
    else
      uiId2 = 65536 + ulIc2 - ic;
  }
}
```

5）TIM3 处理程序设计

TIM3 处理程序设计如下：

```
void TIM3_Proc(void)
//void TIM3_IRQHandler(void)
{
  if(TIM_GetFlagStatus(TIM3, TIM_FLAG_CC1))
  {
    ulOc1 += uiId1*ucDiv>>1;
    if(ulOc1 > 65535) ulOc1 -= 65536;
    TIM_SetCompare1(TIM3, ulOc1);
    TIM_ClearFlag(TIM3, TIM_FLAG_CC1);
  }
  if(TIM_GetFlagStatus(TIM3, TIM_FLAG_CC2))
  {
```



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
    ulOc2 += uiId2/ucMul>>1;
    if(ulOc2 > 65535) ulOc2 -= 65536;
    TIM_SetCompare2(TIM3, ulOc2);
    TIM_ClearFlag(TIM3, TIM_FLAG_CC2);
  }
}
```

6）SysTick 中断处理程序设计

SysTick 中断处理程序设计如下：

```
void SysTick_Handler(void)
{
  ulTick_ms++;
  if(ulTick_ms%1000 == 0)  ulSec++;
}
```

### 6.4.4 系统实现

系统实现的主要步骤如下：

（1）将“207_TIM”文件夹复制并重命名为“304_082”文件夹，在“304_082”文件夹中双击工程文件“STM32.uvproj”打开工程。

（2）修改“adc.c”中的 ADC1_Init_AKEY()程序代码。

（3）在“tim.h”中添加下列函数声明：

```
void TIM2_ICInit_082(void);
void TIM3_OCInit_082(void);
```

（4）在“tim.c”中将 TIM2_ICInit()函数代码复制并重命名为 TIM2_ICInit_082()函数代码。

（5）修改 TIM2_ICInit_082()程序代码。

（6）在“tim.c”中将 TIM1_OCInit()函数代码复制并重命名为 TIM3_OCInit_082()函数代码。

（7）修改 TIM3_OCInit_082()程序代码。

（8）修改文件“main.c”的内容。

（9）在 main()后追加 KEY_Proc()、LED_Proc()、LCD_Proc()、TIM2_Proc()和 TIM3_Proc()的程序代码。

（10）TIM2 捕捉也可以使用中断方式，具体实现步骤如下。

① 在 TIM2_ICInit_082()中添加下列语句：

```
NVIC_InitTypeDef NVIC_InitStruct;

TIM_ITConfig(TIM2, TIM_IT_CC2 | TIM_IT_CC3, ENABLE);
NVIC_InitStruct.NVIC_IRQChannel = TIM2_IRQn;
NVIC_InitStruct.NVIC_IRQChannelPreemptionPriority = 0;
NVIC_InitStruct.NVIC_IRQChannelSubPriority = 0;
NVIC_InitStruct.NVIC_IRQChannelCmd = ENABLE;
NVIC_Init(&NVIC_InitStruct);
```

② 注释掉 while()中的下列语句：

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
//TIM2_Proc();
```

③ 将下列函数名：

```
void TIM2_Proc(void)
```

修改为 TIM2 中断服务程序名：

```
void TIM2_IRQHandler(void)
```

### 6.4.5 客观题

（1）$N$ 个触发器构成的计数器中，有效状态最多有（　　）个。

A．$2^N$　　B．$N$

C．$2N-1$　　D．$2N$

（2）STM32F103RBT6 可采用（　　）位来编辑中断优先级。

A．4　　B．8

C．16　　D．32

（3）欲提高电压比较器的抗干扰能力，应选用下列哪种类型的比较器？（　　）

A．双限比较器　　B．滞回比较器

C．过零比较器　　D．单限比较器

（4）以下哪些可以作为 STM32F103RBT6 片内 RTC 的时钟源？（　　）

A．HSE　　B．LSI

C．HSE/128　　D．LSE

E．HSI　　F．PLL

（5）关于 STM32F103RBT6 微控制器错误的说法是（　　）。

A．IO 口寄存器需以 32 位字的方式访问，但每个 IO 口可以自由编程

B．部分 IO 口兼容 5V 是指 IO 口具有输入、输出 5V 电平的能力

C．片内 ADC 是 12 位的逐次逼近型模拟数字转换器

D．片内 ADC 转换数据对齐方式是可配置的

（6）STM32F103RBT6 微控制器具有_____个 ADC 通道，可以测量_____个外部和_____个内部信号源，SysTick 是一个_____位的倒计数定时器。

（7）图 6.29 所示输入电路中，输入信号 $U_A$、$U_B$ 与输出信号 $U_O$ 之间的关系是（　　）。二极管的正向导通压降为 0.3V，当 $U_A = U_B = 3V$ 时，$U_O$ 为_____V。

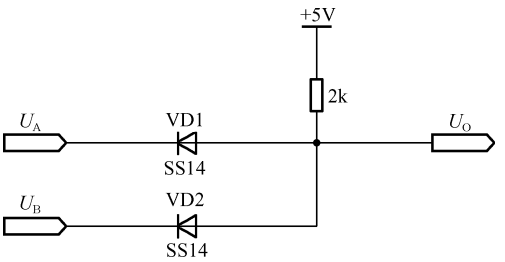



图 6.29　输入电路

A．$U_O = U_A \,\&\, U_B$　　B．$U_O = \overline{U_A | U_B}$

C．$U_O = U_A \mid U_B$　　D．$U_O = \overline{U_A \& U_B}$

（8）对结构体变量 sh 成员引用错误的是（　　）。



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
struct shape{
float length;
int width;
int heigth;
}sh, *ptr;
ptr = &sh;
```

A．sh.width　　B．(*ptr).length

C．ptr→width　　D．ptr.length

（9）图 6.30 所示运放电路中的运放为理想运放，电路的输出电压 $U_O$ 为_____V。

（10）下列关于 Cortex M3 内核与 STM32 控制器的说法中错误的有（　　）。

A．Cortex M3 系列处理器支持 Thumb 指令集

B．Cortex M3 可以支持对单一的比特位进行读写的操作

C．串口既可以工作在异步模式下，也可以工作在同步模式下

D．USART1、USART2 和 USART3 的数据收发均支持 DMA 操作

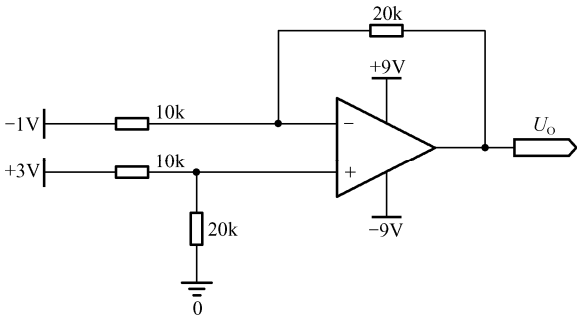



图 6.30　运放电路

参考答案与评分标准

（1）A（2 分）　（2）A（2 分）　（3）B（2 分）

（4）BCD（5 分，每漏选 1 项-2.5 分，多选不得分）　（5）B（2 分）

（6）18，16，2，24（4 分）　（7）A，3.3（5 分，选择 2 分，填空 3 分）

（8）D（2 分）　（9）8（4 分）　（10）A（2 分）

## 6.5　第九届预赛试题——电子定时器

电子定时器方框图如图 6.31 所示。

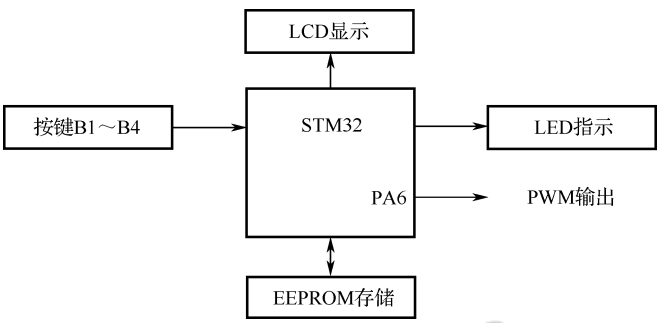



图 6.31　电子定时器方框图

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

电子定时器通过按键设置定时时间，启动定时器后，开始倒计时；计时过程中，可以暂停、取消定时器。在定时时间内，按要求输出 PWM 信号和控制 LED 指示灯。

电子定时器功能描述如下：

1）LCD 显示

LCD 显示界面参考图如图 6.32 所示，包括存储位置、定时时间和当前状态。系统预留 5 个存储位置用于存储常用的定时时间。当定时器停止时，当前状态为 Standby；当系统正在设置时间时，当前状态为 Setting；当定时器运行时，当前状态为 Running，定时器暂停时，当前状态为 Pause。

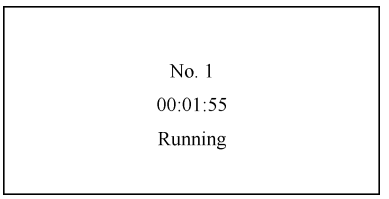



图 6.32　LCD 显示界面参考图

2）按键功能

系统使用 4 个按键：B1、B2、B3 和 B4。

按键 B1 为存储位置切换键。每按一次，存储位置依次以 1、2、3、4、5 循环切换，切换后定时时间设定为当前位置存储的时间。

按键 B2 为时间位置（时、分、秒）切换键和存储键。短按 B2 键进入时间设置状态。每次短按 B2 键，设置位置以时、分、秒循环切换，并突出显示（高亮）当前位置；设置完后，长按 B2 键（超过 0.8s）把设置的时间存储到当前的存储位置，并退出设置状态。如果是临时设置定时时间，则不需存储，直接按定时器启动按键。

按键 B3 为时、分、秒（按键 B2 确定当前位置）数字增加键。每短按 B3 键一次，数字递增一次；按住 B3 键超过 0.8s，则数字快速递增，直到松开。数字递增时，超出范围则从头循环。

按键 B4 为定时器启动键。短按 B4 键，定时器启动，开始运行；运行期间短按 B4 键，暂停定时器，再短按 B4 键，恢复定时器运行；长按 B4 键（超过 0.8s），则取消定时器运行，回到 Standby 状态。

3）PWM 输出和 LED 显示

定时器运行时，PA6 口输出 PWM 信号，同时 LED 灯（LD1）以 1Hz 的频率闪烁。PWM 信号频率为 1kHz，占空比为 80%。

定时器停止或暂停时，停止输出 PWM 信号，LED 灯灭。

4）定时时间存储

设定好的定时时间存储在 EEPROM 中。

掉电重启后显示存储位置 1 的定时时间。

### 6.5.1　系统设计

通过分析系统功能，可以得到电子定时器详细方框图如图 6.33 所示。

PWM 输出由 TIM3 CH1（PA6）实现，输出 1kHz 时分频值为 72 000（大于 65 536），设预分频器的值为 72，则周期值为 1000，匹配值为 800。

定时器主程序流程图如图 6.34 所示。

主程序首先对系统进行初始化，包括 SysTick 初始化、按键接口初始化、LED 接口初始化、LCD 接口初始化、I2C 接口初始化和 TIM3 初始化等。系统初始化后从 AT24C02 中读取存储位置



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

1 的定时时间并显示。

主循环的操作包括按键处理和 LCD 处理，按键处理中包含读取参数、保存参数和允许 PA6 输出，LED 处理和禁止 PA6 输出在 SysTick 中断处理程序中实现。

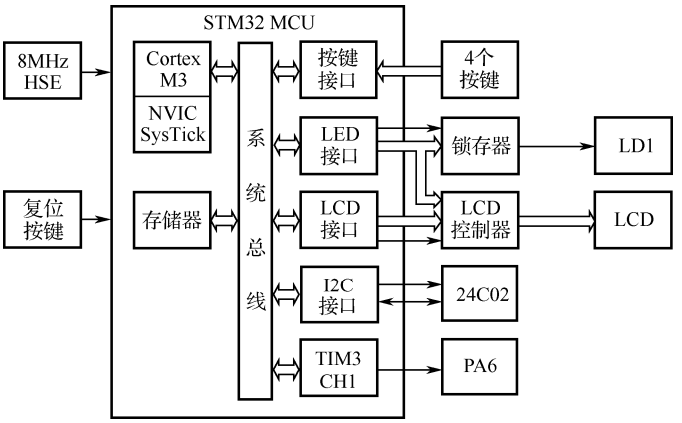



图 6.33　电子定时器详细方框图

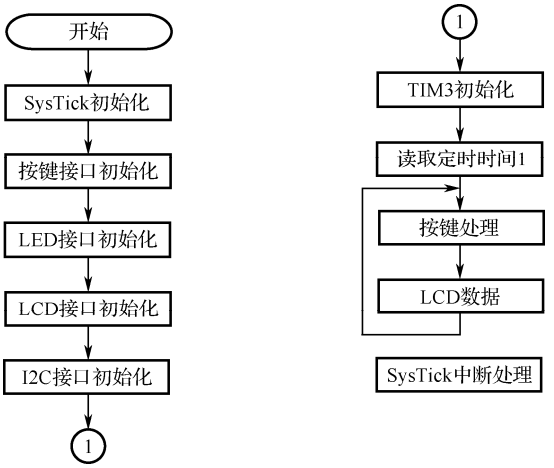



图 6.34　电子定时器主程序流程图

主程序设计如下：

```
#include "key.h"
#include "led.h"
#include "lcd.h"
#include "stdio.h"
#include "i2c.h"
#include "tim.h"

unsigned char ucNo = 1, ucLed = 1;
unsigned char ucHour, ucMin, ucSec;
unsigned char ucState, pucStr[21], ucKey_Long;
unsigned long ulTick_ms, ulSec, ulKey_Time;

void KEY_Proc(void);
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
void LCD_Proc(void);
void B3_Proc(void);

int main(void)
{
  SysTick_Config(72000);
  KEY_Init();
  LED_Init();

  STM3210B_LCD_Init();
  LCD_Clear(Blue);
  LCD_SetBackColor(Blue);
  LCD_SetTextColor(White);

  i2c_init();
  TIM3_OCInit_091();

  i2c_read(pucStr, 0, 3);
  ucHour = pucStr[0];
  ucMin = pucStr[1];
  ucSec = pucStr[2];
  if(ucHour > 24) ucHour = 0;
  if(ucMin > 60) ucMin = 0;
  if(ucSec > 60) ucSec = 0;

  while(1)
  {
    KEY_Proc();
    LCD_Proc();
  }
}
```

### 6.5.2 初始化程序设计

初始化程序包括 SysTick 初始化、按键接口初始化、LED 接口初始化、LCD 接口初始化、I2C 接口初始化和 TIM3 初始化等。

**1）SysTick 初始化程序设计**

SysTick 初始化用 SysTick_Config()实现。

**2）按键接口初始化程序设计**

按键接口初始化程序使用“key.c”中的 KEY_Init()实现。

**3）LED 接口初始化程序设计**

LED 接口初始化程序使用“led.c”中的 LED_Init()实现。

**4）LCD 接口初始化程序设计**

LCD 接口初始化程序使用“lcd.c”中的 STM3210B_LCD_Init()实现。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

5）I2C 接口初始化程序设计

I2C 接口初始化程序使用“i2c.c”中的 i2c_init()实现。

6）TIM3 初始化程序设计

TIM3 初始化程序在“tim.c”中 TIM1_OCInit()的基础上修改实现，内容如下：

```
void TIM3_OCInit_091(void)
{
  GPIO_InitTypeDef GPIO_InitStruct;
  TIM_TimeBaseInitTypeDef TIM_TimeBaseInitStruct;
  TIM_OCInitTypeDef  TIM_OCInitStruct;

  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
  RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3, ENABLE);

  GPIO_InitStruct.GPIO_Pin = GPIO_Pin_6;
  GPIO_InitStruct.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_InitStruct.GPIO_Mode = GPIO_Mode_AF_PP;
  GPIO_Init(GPIOA, &GPIO_InitStruct);

  TIM_TimeBaseInitStruct.TIM_Prescaler = 71;
  TIM_TimeBaseInitStruct.TIM_CounterMode = TIM_CounterMode_Up;
  TIM_TimeBaseInitStruct.TIM_Period = 999;
  TIM_TimeBaseInitStruct.TIM_ClockDivision = TIM_CKD_DIV1;
  TIM_TimeBaseInitStruct.TIM_RepetitionCounter = 0;
  TIM_TimeBaseInit(TIM3, &TIM_TimeBaseInitStruct);

  TIM_OCInitStruct.TIM_OCMode = TIM_OCMode_PWM1;
  TIM_OCInitStruct.TIM_OutputState = TIM_OutputState_Disable;
  TIM_OCInitStruct.TIM_Pulse = 800;
  TIM_OCInitStruct.TIM_OCPolarity = TIM_OCPolarity_High;
  TIM_OC1Init(TIM3, &TIM_OCInitStruct);

  TIM_Cmd(TIM3, ENABLE);
}
```

### 6.5.3 处理程序设计

处理程序包括按键处理、LCD 处理和 SysTick 中断处理等，程序流程图如图 6.35 所示。

1）按键处理程序设计

按键处理程序根据状态图进行设计。电子定时器工作状态图如图 6.36 所示。

系统共有 6 个状态，其中 3 个设置状态，分别设置时、分和秒。待机时用 B1 键切换定时值，短按 B2 键进入设置状态，并在设置状态间切换，短按 B4 键进入运行状态，并在运行和暂停状态间切换；设置状态下按 B3 键增加设置值，长按 B2 键保存设置值并返回待机状态，短按 B4 键进入运行状态；运行或暂停状态时长按 B4 键返回待机状态，定时结束时也返回待机状态。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

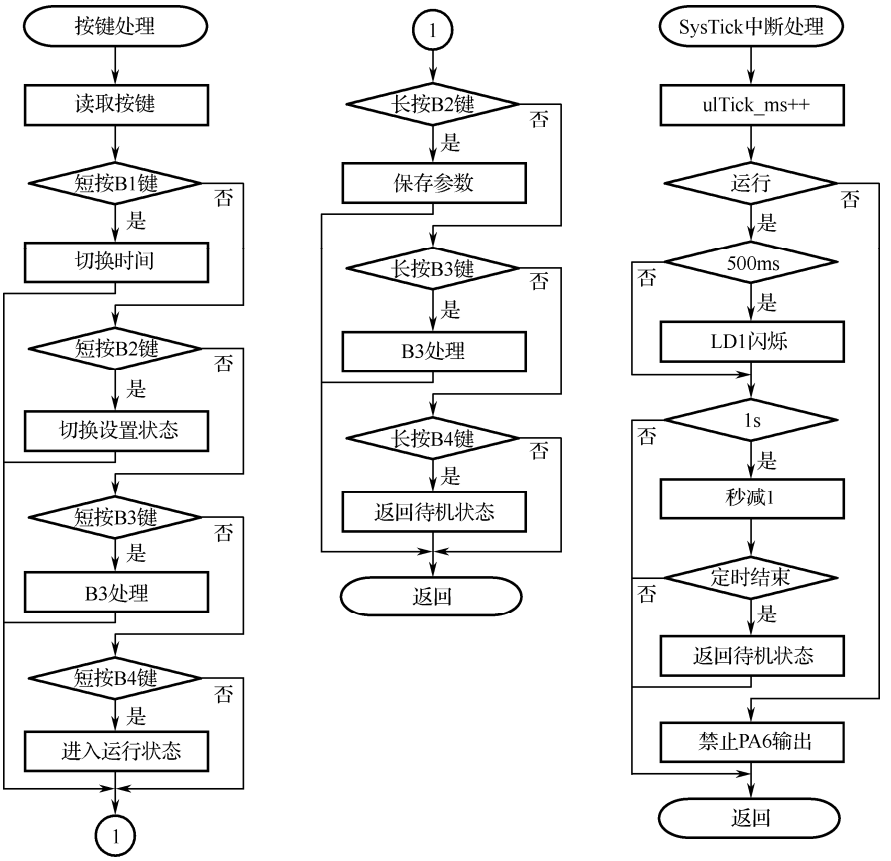



图 6.35　电子定时器处理程序流程图

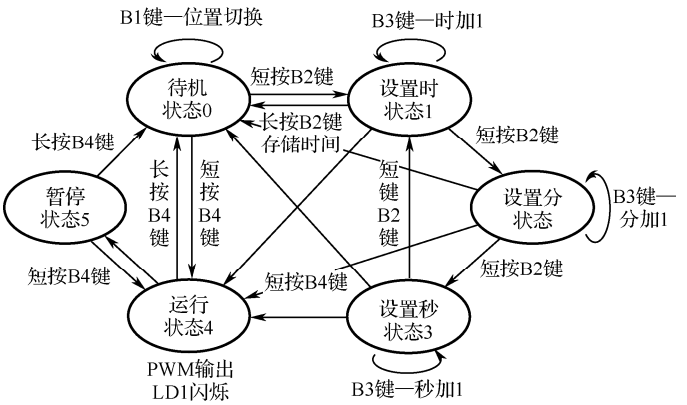



图 6.36　电子定时器工作状态图

按键处理程序设计如下：

```
void KEY_Proc(void)
{
  unsigned char ucKey_Val;

  ucKey_Val = KEY_Scan();
  if(ucKey_Val != ucKey_Long)
  {
    ucKey_Long = ucKey_Val;
```



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
  ulKey_Time = ulTick_ms;
}
else
  ucKey_Val = 0;

switch(ucKey_Val)
{
  case 1:                              // B1: 存储位置切换键
    if(!ucState)
    {
      if(++ucNo == 6)
        ucNo = 1;
      i2c_read(pucStr, 3*(ucNo-1), 3);
      ucHour = pucStr[0];
      ucMin  = pucStr[1];
      ucSec  = pucStr[2];
    }
    break;
  case 2:                              // B2: 时间位置切换键
    if(!ucState)
      ucState = 1;
    else
      if(ucState < 4)
        if(++ucState == 4)
          ucState = 1;
    break;
  case 3:                              // B3: 时间加
    B3_Proc();
    break;
  case 4:                              // B4: 启动键
    if(ucState < 4)
      ulSec = (long)ucHour*3600 + ucMin*60 + ucSec;
    if(ucState != 4)
    {
      ucState = 4;
      TIM_CCxCmd(TIM3, TIM_Channel_1, ENABLE);
    }
    else
      ucState = 5;
}
// 长按键处理
if(ulTick_ms-ulKey_Time > 800)
{
  switch(ucKey_Long)
  {
    case 2:                        // 长按 B2
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
        if(ucState == 1 || ucState == 2 || ucState == 3)
        {
          pucStr[0] = ucHour;
          pucStr[1] = ucMin;
          pucStr[2] = ucSec;
          i2c_write(pucStr, 3*(ucNo - 1), 3);

          ucState = 0;
        }
        break;
      case 3:                       // 长按 B3
        B3_Proc();
        break;
      case 4:                       // 长按 B4
        if(ucState == 4 || ucState == 5)
          ucState = 0;
    }
  }
}

void B3_Proc(void)
{
  switch(ucState)
  {
    case 1:
      if(++ucHour > 23)
        ucHour = 0;
      break;
    case 2:
      if(++ucMin > 59)
        ucMin = 0;
      break;
    case 3:
      if(++ucSec > 59)
        ucSec = 0;
  }
}
```

2）LCD 处理程序设计

LCD 处理程序设计如下：

```
void LCD_Proc(void)
{
  sprintf((char*)pucStr, "       No.%1u", ucNo);
  LCD_DisplayStringLine(Line3, pucStr);

  if(ucState == 1) LCD_SetBackColor(Red);
```



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
  LCD_DisplayChar(Line5, 224, ucHour/10+0x30);
  LCD_DisplayChar(Line5, 208, ucHour%10+0x30);
  LCD_SetBackColor(Blue);
  LCD_DisplayChar(Line5, 192, ':');

  if(ucState == 2) LCD_SetBackColor(Red);
  LCD_DisplayChar(Line5, 176, ucMin/10+0x30);
  LCD_DisplayChar(Line5, 160, ucMin%10+0x30);
  LCD_SetBackColor(Blue);
  LCD_DisplayChar(Line5, 144, ':');

  if(ucState == 3) LCD_SetBackColor(Red);
  LCD_DisplayChar(Line5, 128, ucSec/10+0x30);
  LCD_DisplayChar(Line5, 112, ucSec%10+0x30);
  LCD_SetBackColor(Blue);

  switch(ucState)
  {
    case 0:
      LCD_DisplayStringLine(Line7, (u8*)"     Standby.");
      break;
    case 1:
    case 2:
    case 3:
      LCD_DisplayStringLine(Line7, (u8*)"     Setting.");
      break;
    case 4:
      LCD_DisplayStringLine(Line7, (u8*)"     Running.");
      break;
    case 5:
      LCD_DisplayStringLine(Line7, (u8*)"      Pause. ");
  }
}
```

### 3）SysTick 中断处理程序设计

SysTick 中断处理程序设计如下：

```
void SysTick_Handler(void)
{
  ulTick_ms++;

  if(ucState == 4)
  {
    if(ulTick_ms%500 == 0)
    {
      ucLed ^= 1;
      LED_Disp(ucLed);
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
    }
    if(ulTick_ms%1000 == 0)
    {
      if(ulSec--)
      {
        ucHour = ulSec / 3600;
        ucMin  = ulSec % 3600 / 60;
        ucSec  = ulSec % 3600 % 60;
      }
      else
        ucState = 0;
    }
  }
  else
  {
    TIM_CCxCmd(TIM3, TIM_Channel_1, DISABLE);
    LED_Disp(0);
  }
}
```

### 6.5.4 系统实现

系统实现的主要步骤如下：

（1）将“207_TIM”文件夹复制并重命名为“305_091”文件夹，在“305_091”文件夹中双击工程文件“STM32.uvproj”打开工程。

（2）在“tim.h”中添加下列函数声明：

```
void TIM3_OCInit_091(void);
```

（3）在“tim.c”中将 TIM1_OCInit()函数代码复制并重命名为 TIM3_OCInit_091()函数代码。

（4）修改 TIM3_OCInit_091()函数代码。

（5）修改文件“main.c”的内容。

（6）在 main()后追加 KEY_Proc()、B3_Proc()和 LCD_Proc()的程序代码。

（7）修改 SysTick_Handler()的程序代码。

### 6.5.5 客观题

（1）STM32F103RBT6 微控制器中 APB1 总线的最高运行频率是_____MHz。

（2）STM32F103RBT6 微控制器中 ADC 的分辨率是_____位，通道数共有_____个，其中外部通道数为_____个。

（3）函数 $F=\overline{A}B+AC+ABCD+\overline{A}$ 的最简式为（　　）。

A．C＋D　　B．C

C．$\overline{B}$+C　　D．$\overline{A}$+C

（4）当温度升高时，二极管的反向饱和电流将（　　）。

A．增大　　B．减小

C．保持不变　　D．与温度没有直接关系



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

（5）模数转换器 ADC 的主要技术指标有（　　）。

A．量化误差　　B．转换速率

C．分辨率　　D．频率

（6）Cortex M3 处理器是（　　）级流水线。

A．2　　B．5

C．4　　D．3

（7）设计一个 8421 BCD 码计数器至少需要（　　）个触发器。

A．4　　B．5

C．8　　D．3

（8）为了使高阻抗信号源与低阻抗负载进行配合，在设计电路过程中往往需要进行阻抗匹配，以下哪种电路适合接入高阻抗信号源与低阻抗负载之间？（　　）

A．共射电路　　B．共基电路

C．共集电路　　D．以上都可以

（9）图 6.37 所示运放电路中，输入与输出的关系是（　　）。

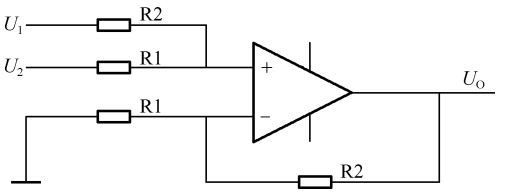



图 6.37　运放电路

A．$Uo=U_1+U_2R1/R2$　　B．$Uo=U_2+U_1R1/R2$

C．$Uo=U_1+U_2R2/R1$　　D．$Uo=U_2+U_1R2/R1$

（10）在 STM32 处理器中，以下哪种时钟源可被用来驱动系统时钟 SYSCLK？（　　）

A．HSE　　B．HSI

C．LSI　　D．PLL

参考答案与评分标准

（1）36（3 分）　（2）12，18，16（3 分）　（3）D（3 分）　（4）A（3 分）　（5）ABC（3 分）

（6）D（3 分）　（7）A（3 分）　（8）C（3 分）　（9）C（3 分）　（10）ABD（3 分）

## 6.6　第九届决赛试题——电子秤

电子秤方框图如图 6.38 所示。

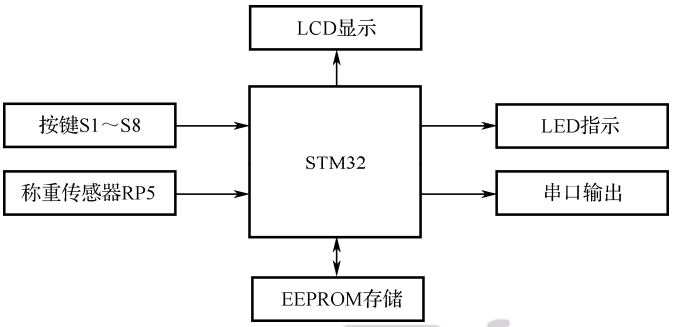



图 6.38　电子秤方框图

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

电子秤功能描述如下。

**1）基本功能**

（1）通过资源扩展板电位器 RP5 模拟称重传感器输出信号，STM32 采集此电压信号，并计算货物重量。电子秤称重范围为 0～10 kg。

（2）通过按键选择不同货物、计费及货物单价参数设置等功能。

（3）通过 LCD 显示所称货物重量和货物价格等信息。

（4）设备基本工作流程：选择货物（按键操作）——称重——计费（按键操作）——更新显示及串口输出。

**2）显示功能**

（1）货物重量及价格显示如图 6.39 所示。

（2）货物单价设置显示如图 6.40 所示。

① 货物单价、重量和总价保留小数点后两位有效数字。

② 设置次数在每次设置完成后加 1，如未改动任何货物单价则不累加。

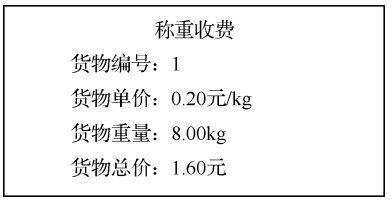



图 6.39　货物重量及价格显示

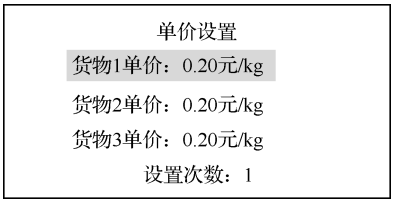



图 6.40　货物单价设置显示

**3）按键功能**

（1）嵌入式资源扩展板按键功能定义如图 6.41 所示。

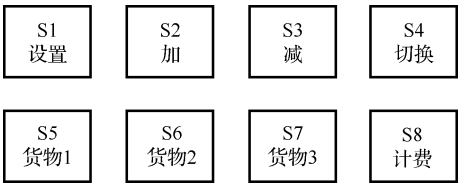



图 6.41　嵌入式资源扩展板按键功能定义

（2）按键功能说明。

- 设置：按下“设置”按键后，进入单价设置界面，通过“切换”按键切换待调整的货物单价，被选择的显示单元高亮显示，然后可以通过“加”和“减”按键进行调整，再次按下“设置”按键保存本次的调整结果到 EEPROM 存储器，退出单价设置界面，返回称重收费界面，并将参数配置信息通过串口输出。

串口配置信息输出格式如下：

U.W.1: 0.24　　--U.W.1 代表货物 1 单价

U.W.2: 0.20　　--U.W.2 代表货物 2 单价

U.W.3: 0.20　　--U.W.3 代表货物 3 单价

① “加”、“减”和“切换”按键仅在设置界面下有效，货物单价步进单位为 0.01 元。长按“加”或“减”按键 0.8s 以上，可实现快速加或减功能。

② 货物单价设置范围为 0～10 元。

- 货物 1 ~ 货物 3：选择不同货物
- 计费：按下此按键，LCD 更新计费结果，并通过串口输出计费数据信息。

串口计费信息输出格式如下：

U.W.1: 0.24　　--U.W.1 代表货物 1 单价

G.W: 5.00　　--G.W 代表货物重量

Total: 1.20　　--Total 代表总价



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

**4）存储功能**

通过 EEPROM 实现货物单价和设置次数的存储功能，设备重新上电后能够自动从 EEPROM 中载入全部参数。

**5）称重功能**

通过资源扩展板上的电位器 RP5 模拟称重传感器的输出信号，0～3.3V 对应货物重量 0～10kg（线性关系），称重结果保留小数点后两位有效数字。

**6）LED 指示功能**

称重计费状态下，LD1 以 0.8 s 为间隔亮灭。

单价设置状态下，LD1 以 0.4 s 为间隔亮灭。

**7）串口功能**

使用实训平台上的 USART2（USB 转串口）完成串口输出功能。

串口通信波特率：9600 bps。

**8）资源扩展板跳线配置**

资源扩展板跳线配置如图 6.42 所示。

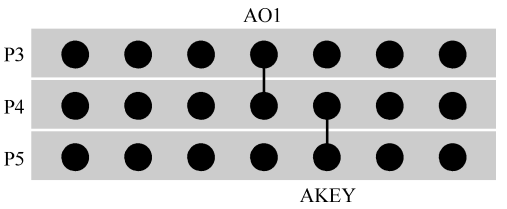



图 6.42　资源扩展板跳线配置

## 6.6.1　系统设计

通过分析系统功能，可以得到电子秤详细方框图如图 6.43 所示。

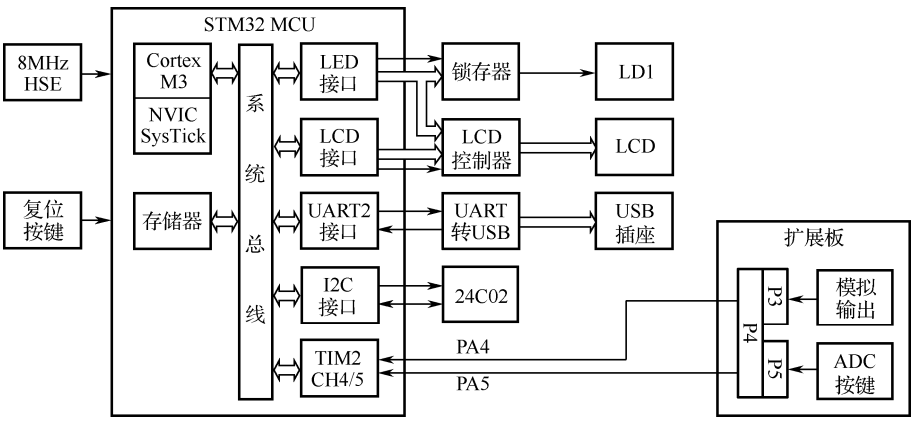



图 6.43　电子秤详细方框图

扩展板模拟输出 AO1 模拟称重传感器输出信号，通过 PA4（ADC_IN4）输入 STM32；扩展板 ADC 按键输出 AKEY 通过 PA5（ADC_IN5）输入 STM32。2 路模拟信号可以使用 2 路规则通道实现模数转换（必须使用 DMA），也可以 1 路使用规则通道，另 1 路使用注入通道。这里使用第 2 种方法。

电子秤主程序流程图如图 6.44 所示。主程序首先对系统进行初始化，包括 SysTick 初始化、LED 接口初始化、LCD 接口初始化、UART 接口初始化、I2C 接口初始化和 ADC 初始化等。系统初始化后从 AT24C02 中读取并设置参数。

主循环的操作包括按键处理、LCD 处理、UART 处理和 ADC 处理，按键处理中包含保存参数，ADC 处理实现称重，LED 处理在 SysTick 中断处理程序中实现。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

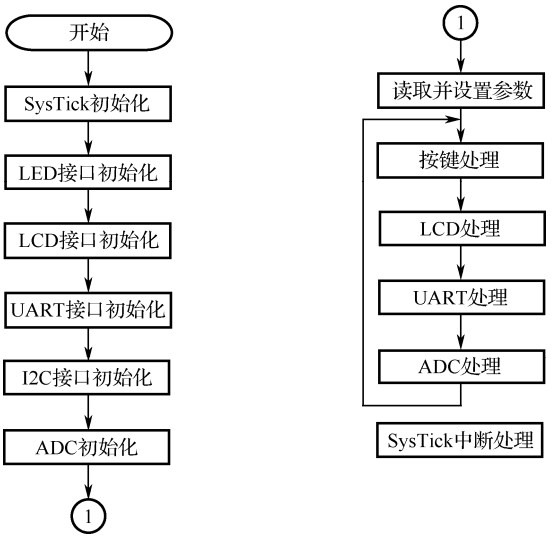



图 6.44　电子秤主程序流程图

主程序设计如下：

```
#include "led.h"
#include "lcd.h"
#include "usart.h"
#include "i2c.h"
#include "adc.h"

unsigned int  puiUw[3], puiUw_Old[3], uiGw, uiGt;
unsigned char ucNo = 1, ucCnt, ucLed, ucUart;
unsigned char ucState, pucStr[21];
unsigned char ucSec, ucSec1, ucAkey_Long;
unsigned long ulTick_ms, ulAkey_Time;

void Para_Read(void);
void AKEY_Proc(void);
void LCD_Proc(void);
void UART_Proc(void);
void ADC_Proc(void);
void S1_Proc(void);
void S2_Proc(void);
void S3_Proc(void);

int main(void)
{
  SysTick_Config(72000);
  LED_Init();

  STM3210B_LCD_Init();
  LCD_Clear(Blue);
```



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
  LCD_SetBackColor(Blue);
  LCD_SetTextColor(White);

  USART2_Init(9600);
  i2c_init();
  ADC1_Init_AKEY();

  Para_Read();

  while(1)
  {
    AKEY_Proc();
    LCD_Proc();
    UART_Proc();
    ADC_Proc();
  }
}

void Para_Read(void)
{
  i2c_read(pucStr, 0, 8);
  if(pucStr[0] == 0xaa)
  {
    puiUw[0] = pucStr[2]*100 + pucStr[1];
    puiUw[1] = pucStr[4]*100 + pucStr[3];
    puiUw[2] = pucStr[6]*100 + pucStr[5];
    ucCnt = pucStr[7];
  }
  else
  {
    puiUw[0] = 100;
    puiUw[1] = 200;
    puiUw[2] = 300;
    ucCnt = 0;
  }
}
```

### 6.6.2 初始化程序设计

初始化程序包括 SysTick 初始化、LED 接口初始化、LCD 接口初始化、UART 接口初始化、I2C 接口初始化和 ADC 初始化等。

**1）SysTick 初始化程序设计**

SysTick 初始化用 SysTick_Config()实现。

**2）LED 接口初始化程序设计**

LED 接口初始化程序使用“led.c”中的 LED_Init()实现。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

3）LCD 接口初始化程序设计

LCD 接口初始化程序使用“lcd.c”中的 STM3210B_LCD_Init()实现。

4）UART 接口初始化程序设计

UART 接口初始化程序使用“usart.c”中的 USART2_Init()实现。

5）I2C 接口初始化程序设计

I2C 接口初始化程序使用“i2c.c”中的 i2c_init()实现。

6）ADC 初始化程序设计

ADC 的通道 4（PA4）用于称重，用注入通道实现；通道 5（PA5）用于按键，用规则通道实现。ADC 初始化程序在“adc.c”中 ADC1_Init_AKEY()的基础上修改实现，内容如下：

```
void ADC1_Init_AKEY(void)
{
  GPIO_InitTypeDef GPIO_InitStruct;
  ADC_InitTypeDef ADC_InitStruct;

  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1, ENABLE);

  GPIO_InitStruct.GPIO_Pin = GPIO_Pin_4 | GPIO_Pin_5;
  GPIO_InitStruct.GPIO_Mode = GPIO_Mode_AIN;
  GPIO_Init(GPIOA, &GPIO_InitStruct);

  ADC_InitStruct.ADC_Mode = ADC_Mode_Independent;
  ADC_InitStruct.ADC_ScanConvMode = DISABLE;
  ADC_InitStruct.ADC_ContinuousConvMode = DISABLE;
  ADC_InitStruct.ADC_ExternalTrigConv = ADC_ExternalTrigConv_None;
  ADC_InitStruct.ADC_DataAlign = ADC_DataAlign_Right;
  ADC_InitStruct.ADC_NbrOfChannel = 1;
  ADC_Init(ADC1, &ADC_InitStruct);

  ADC_RegularChannelConfig(ADC1, ADC_Channel_5, 1,
    ADC_SampleTime_1Cycles5);
  ADC_InjectedChannelConfig(ADC1, ADC_Channel_4, 1,
    ADC_SampleTime_1Cycles5);
  ADC_AutoInjectedConvCmd(ADC1, ENABLE);

  ADC_Cmd(ADC1, ENABLE);
  ADC_StartCalibration(ADC1);
  while(ADC_GetCalibrationStatus(ADC1));
}
```

### 6.6.3 处理程序设计

处理程序包括按键处理、LCD 处理、UART 处理、ADC 处理和 LED 处理等，LED 处理在 SysTick 中断处理程序中实现，处理程序流程图如图 6.45 所示。



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

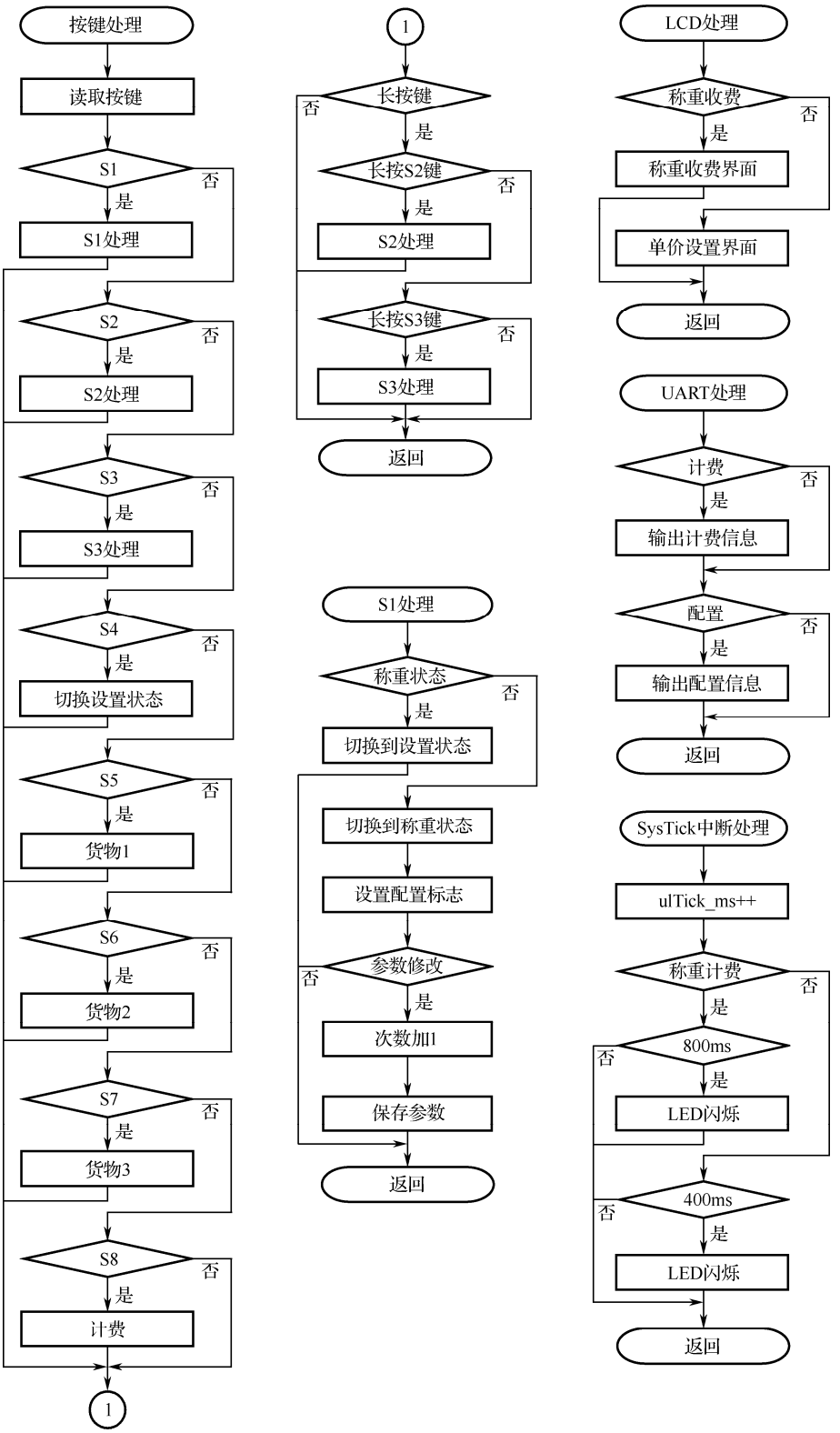



图 6.45　电子秤处理程序流程图

### 1）按键处理程序设计

按键处理程序使用状态图进行设计。电子秤工作状态图如图 6.46 所示。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

系统共有 4 个状态：1 个称重状态和 3 个设置状态（分别设置单价 1～3）。称重状态和设置状态间用 S1 键切换，3 个设置状态间用 S4 键切换，3 个设置状态均可用 S2 或 S3 键加或减参数，S5～S7 键用于选择货物 1～3，S8 键用于称重计费。

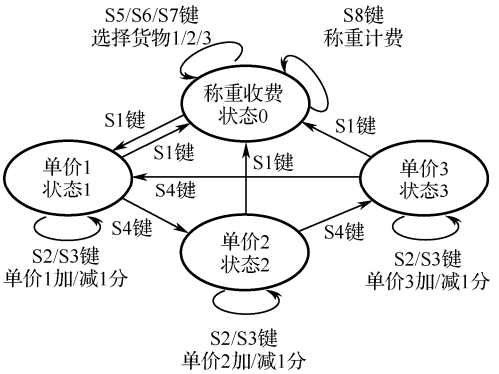



图 6.46　电子秤工作状态图

按键处理程序设计如下：

```
void AKEY_Proc(void)
{
  unsigned char ucAkey_Val;
  ucAkey_Val = AKEY_Read();
  if(ucAkey_Val != ucAkey_Long)
  {
    ucAkey_Long = ucAkey_Val;
    ulAkey_Time = ulTick_ms;
  }
  else
    ucAkey_Val = 0;

  switch(ucAkey_Val)
  {
    case 1:                              // S1: 设置键
      S1_Proc();  break;
    case 2:                              // S2: 参数加
      S2_Proc();  break;
    case 3:                              // S3: 参数减
      S3_Proc();  break;
    case 4:                              // S4: 切换键
      if(ucState)
        if(++ucState == 4)               // 切换货物 1～3
          ucState = 1;
      break;
    case 5:                              // S5: 货物 1
      if(!ucState)  ucNo = 1;  break;
    case 6:                              // S6: 货物 2
      if(!ucState)  ucNo = 2;  break;
```



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
    case 7:                              // S7: 货物 3
      if(!ucState)  ucNo = 3;  break;
    case 8:                              // S8: 计费键
      if(!ucState)
      {
        ucUart = 1;                      // 设置计费标志
        uiGt = uiGw*puiUw[ucNo-1];
      }
  }
  // S2 和 S3 长按处理
  if(ulTick_ms-ulAkey_Time > 800)
  {
    switch(ucAkey_Long)
    {
      case 2:                            // S2: 参数加
        S2_Proc();  break;
      case 3:                            // S3: 参数减
        S3_Proc();
    }
  }
}
// S1 键处理
void S1_Proc(void)
{
  if(!ucState)
  {
    ucState = 1;                         // 切换到设置状态

    puiUw_Old[0] = puiUw[0];
    puiUw_Old[1] = puiUw[1];
    puiUw_Old[2] = puiUw[2];
  }
  else
  {
    ucState = 0;                         // 切换到称重状态
    ucUart = 2;                          // 设置配置标志

    if((puiUw[0] != puiUw_Old[0]) || (puiUw[1] != puiUw_Old[1])
      || (puiUw[2] != puiUw_Old[2]))
    {
      pucStr[0] = 0xaa;
      pucStr[1] = puiUw[0] % 100;
      pucStr[2] = puiUw[0] / 100;
      pucStr[3] = puiUw[1] % 100;
      pucStr[4] = puiUw[1] / 100;
      pucStr[5] = puiUw[2] % 100;
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
      pucStr[6] = puiUw[2] / 100;
      pucStr[7] = ++ucCnt;
      i2c_write(pucStr, 0, 8);        // 保存参数
    }
  }
}
// S2 键处理
void S2_Proc(void)
{
  if(ucState)
    if(puiUw[ucState-1] < 1000)
      puiUw[ucState-1]++;             // 单价加 1 分
}
// S3 键处理
void S3_Proc(void)
{
  if(ucState)
    if(puiUw[ucState-1] > 0)
      puiUw[ucState-1]--;             // 单价减 1 分
}
```

2）LCD 处理程序设计

LCD 处理程序设计如下：

```
void LCD_Proc(void)
{
  float temp;

  if(!ucState)                        // 称重收费
  {
    sprintf((char*)pucStr, "  Weight & Charge   ");
    LCD_DisplayStringLine(Line1, pucStr);

    sprintf((char*)pucStr, "Number:  %1u         ", ucNo);
    LCD_DisplayStringLine(Line3, pucStr);

    temp = (float)puiUw[ucNo-1]/100;
    sprintf((char*)pucStr, "Price:  %5.2fYuan/kg", temp);
    LCD_DisplayStringLine(Line5, pucStr);

    temp = (float)uiGw/1000;
    sprintf((char*)pucStr, "Weight: %5.2fkg    ", temp);
    LCD_DisplayStringLine(Line7, pucStr);

    temp = (float)uiGt/100000;
    sprintf((char*)pucStr, "Total:  %5.2fYuan   ", temp);
    LCD_DisplayStringLine(Line9, pucStr);
  }
```



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
  else                                    // 设置参数
  {
    sprintf((char*)pucStr, " Parameter Setting  ");
    LCD_DisplayStringLine(Line1, pucStr);

    temp = (float)puiUw[0]/100;
    sprintf((char*)pucStr, "Price 1:%5.2fYuan/kg", temp);
    if(ucState == 1) LCD_SetBackColor(Red);
    LCD_DisplayStringLine(Line3, pucStr);
    LCD_SetBackColor(Blue);

    temp = (float)puiUw[1]/100;
    sprintf((char*)pucStr, "Price 2:%5.2fYuan/kg", temp);
    if(ucState == 2) LCD_SetBackColor(Red);
    LCD_DisplayStringLine(Line5, pucStr);
    LCD_SetBackColor(Blue);

    temp = (float)puiUw[2]/100;
    sprintf((char*)pucStr, "Price 3:%5.2fYuan/kg", temp);
    if(ucState == 3) LCD_SetBackColor(Red);
    LCD_DisplayStringLine(Line7, pucStr);
    LCD_SetBackColor(Blue);

    sprintf((char*)pucStr, "Count:  %2u       ", ucCnt);
    LCD_DisplayStringLine(Line9, pucStr);
  }
}
```

### 3）UART 处理程序设计

UART 处理程序设计如下：

```
void UART_Proc(void)
{
  float temp;

  if(ucUart == 1)
  { // 输出计费信息
    temp = (float)puiUw[ucNo-1]/100;
    printf("U.W.%1u: %5.2f\r\n", ucNo, temp);
    temp = (float)uiGw/1000;
    printf("G.W:   %5.2f\r\n", temp);
    temp = (float)uiGt/100000;
    printf("TOTAL: %5.2f\r\n", temp);
  }
  if(ucUart == 2)
  { // 输出配置信息
    temp = (float)puiUw[0]/100;
    printf("U.W.1: %5.2f\r\n", temp);
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
    temp = (float)puiUw[1]/100;
    printf("U.W.2: %5.2f\r\n", temp);
    temp = (float)puiUw[2]/100;
    printf("U.W.3: %5.2f\r\n", temp);
  }
  ucUart = 0;
}

void ADC_Proc(void)
{
  if(ucSec1 != ucSec)
  {
    ucSec1 = ucSec;
    uiGw = ADC1_InjeConv()*10000/4095;
  }
}
```

4）LED 处理程序设计

LED 处理用 SysTick 中断处理程序实现，程序设计如下：

```
// SysTick 中断处理程序
void SysTick_Handler(void)
{
  ulTick_ms++;
  if(!ucState)
  {
    if(ulTick_ms%800 == 0)
    {
      ucLed ^= 1;
      LED_Disp(ucLed);
    }
  }
  else
  {
    if(ulTick_ms%400 == 0)
    {
      ucLed ^= 1;
      LED_Disp(ucLed);
    }
  }
}
```

### 6.6.4 系统实现

系统实现的主要步骤如下：

（1）将“206_ADC”文件夹复制并重命名为“306_092”文件夹，在“306_092”文件夹中双击工程文件“STM32.uvproj”打开工程。



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

（2）修改“adc.c”中 ADC1_Init_AKEY()的函数代码。

（3）修改文件“main.c”的内容。

（4）在 main()后追加 AKEY_Proc()、S1_Proc()、S2_Proc()、S3_Proc()、LCD_Proc()、UART_Proc()和 ADC_Proc()的程序代码。

（5）修改 SysTick_Handler()的程序代码。

### 6.6.5 客观题

（1）STM32F103RBT6 处理器 TIM1 具有____位可编程预分频器，时钟分频系数为____至____之间的任意值。

（2）由理想运算放大器构成的小信号交流放大电路如图 6.47 所示，通频带内的电压放大倍数为____，下限频率 $f_L$ 为____Hz。（电压放大倍数、截止频率均取整数部分）。

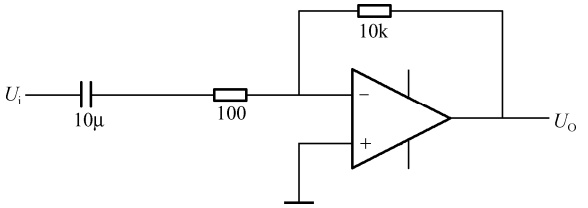



图 6.47　小信号交流放大电路

（3）在 STM32F103 中断向量控制器管理下，可将中断分为（　　）。

A．4 组　　B．5 组

C．6 组　　D．8 组

（4）在 STM32 处理器中一个 DMA 请求，至少占用（　　）个周期的 CPU 访问系统总线时间。

A．1　　B．2

C．3　　D．4

（5）STM32 处理器 APB2 上的 IO 引脚最大翻转速度为（　　）。

A．16MHz　　B．18MHz

C．36MHz　　D．72MHz

（6）两个电压放大倍数相同（电路相同，且采用同一种晶体管）的 A 和 B 电路，对同一个信号源的电压进行放大，在负载开路的条件下，测得 A 电路的输出电压较小，不考虑仪表的测量误差，这说明 A 电路（　　）。

A．输入电阻大　　B．输出电阻大

C．输入电阻小　　D．输出电阻小

（7）ARM-CORTEX M3 微控制器可以通过以下哪些外设唤醒？（　　）

A．IO 端口　　B．RTC 闹钟输出

C．USB 唤醒事件　　D．PLL

（8）测得某些电路中三极管的各极电压如图 6.48 所示，哪些三极管处于放大状态？（　　）

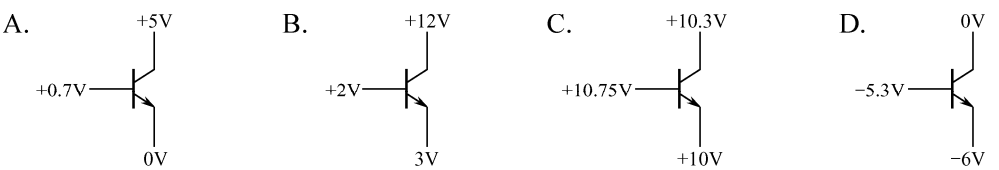



图 6.48　三极管各极电压

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

（9）同步时序电路和异步时序电路比较，其差异在于后者（　　）。

A．没有触发器　　B．没有统一的时钟控制脉冲

C．没有稳定状态　　D．输出只与内部状态有关

（10）以下关于 STM32 处理器的描述中错误的是（　　）。

A．DMA 可编程控制器的数据传输最大数目为 65535

B．中断屏蔽器可以屏蔽所有的异常和中断

C．STM32 ADC 是 12 位的连续近似模拟到数字转换器

D．外设的 DMA 请求通过逻辑“与”关系送到 DMA 控制器，同时可以有多个请求有效

参考答案与评分标准

（1）16，1，65536（3 分），每空 1 分）（2）-100，159（3 分，每空 1.5 分）

（3）B（3 分）（4）B（3 分）（5）B（3 分）（6）C（3 分）（7）ABC（3 分）

（8）AD（3 分）（9）B（3 分）（10）ABD（3 分）

## 6.7　第十届预赛试题

系统方框图如图 6.49 所示（本科生组没有“EEPROM 存储”模块）。

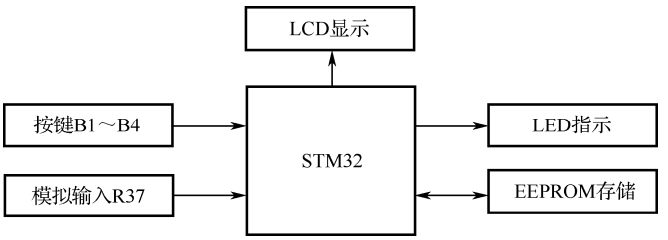



图 6.49　系统方框图

系统功能描述如下：

**1）基本功能**

（1）测量实训平台上电位器 R37 输出的模拟电压 $V_{R37}$，并通过液晶屏实现数据展示。

（2）通过 LED 指示灯实现超出上限、低于下限的提醒功能。

（3）通过按键实现阈值范围和输出提醒指示灯的设置功能。

（4）通过 EEPROM 实现参数的断电存储功能。

**2）显示功能**

（1）数据显示界面。

数据显示界面包含 3 个显示要素：界面名称、采集的实时电压数据和状态，参考界面如图 6.50 所示。

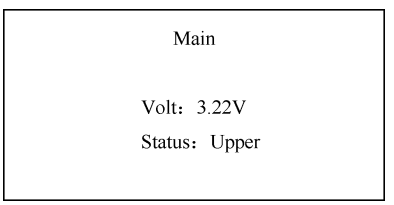



图 6.50　数据显示参考界面

其中状态（Status）包括超过上限（Upper）、低于下限（Lower）和正常（Normal）。



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

（2）参数配置界面。

参数配置界面包含 5 个显示要素：界面名称、电压的上限值、电压的下限值、电压超过上限的提醒指示灯和电压低于下限的提醒指示灯，参考界面如图 6.51 所示。

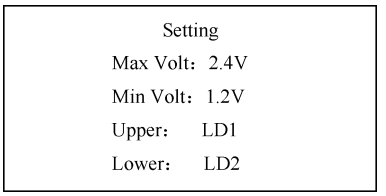



图 6.51　参数配置参考界面

其中电压上下限值设置范围为 0～3.3V，设备应具备错误设置的保护功能。电压提醒指示灯设置范围为 LD1～LD8，设备应禁止将上下限指示灯设置为同一个指示灯。

**3）按键功能**

按键 B1 为设置键，按下此按键，切换选择数据显示界面和参数设置界面，返回数据显示界面时将参数保存到 EEPROM 存储器。

按键 B2 为选择键，在参数设置界面按下此按键，切换选择参数项（电压的上限值、电压的下限值、电压超过上限的提醒指示灯和低于下限的提醒指示灯），被选择的参数项“高亮”显示。

按键 B3 为加键，进行电压参数设置时，当前选择的电压参数加 0.3V。进行 LED 提醒指示灯参数设置时，LED 指示灯序号加 1。

按键 B4 为减键，进行电压参数设置时，当前选择的电压参数减 0.3V。进行 LED 提醒指示灯参数设置时，LED 指示灯序号减 1。

B2、B3 和 B4 键仅在设置界面下有效。

**4）LED 功能**

（1）R37 输出电压值超过电压上限值时（$V_{R37}>V_{上限}$），上限提醒指示灯以 0.2s 为间隔闪烁，下限指示灯熄灭。

（2）R37 输出电压值低于电压下限值时（$V_{R37}<V_{下限}$），下限提醒指示灯以 0.2s 为间隔闪烁，上限指示灯熄灭。

（3）R37 输出电压值介于上限和下限电压之间时（$V_{下限}\leqslant V_{R37}\leqslant V_{上限}$），上限和下限提醒指示灯熄灭。

（4）实训平台上除当前用作超出上限和低于下限提醒功能的指示灯外，其他指示灯应处于熄灭状态。

**5）初始状态说明**

（1）设备重新上电时，应从 EEPROM 中载入上下限提醒指示灯、上限电压和下限电压参数。

（2）默认提醒指示灯：超过上限提醒指示灯为 LD1，低于下限提醒指示灯为 LD2。

（3）默认电压阈值参数：电压的上限值为 2.4V，电压的下限值为 1.2V。

## 6.7.1　系统设计

通过分析系统功能，可以得到系统详细方框图如图 6.52 所示。

系统主程序流程图如图 6.53 所示。

主程序首先对系统进行初始化，包括 SysTick 初始化、按键接口初始化、LED 接口初始化、LCD 接口初始化、I2C 接口初始化和 ADC 初始化等。系统初始化后从 AT24C02 中读取参数并判断有效性。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

主循环的操作包括按键处理、LCD 处理和 ADC 处理，按键处理中包含保存参数，LED 处理在 SysTick 中断处理程序中实现。

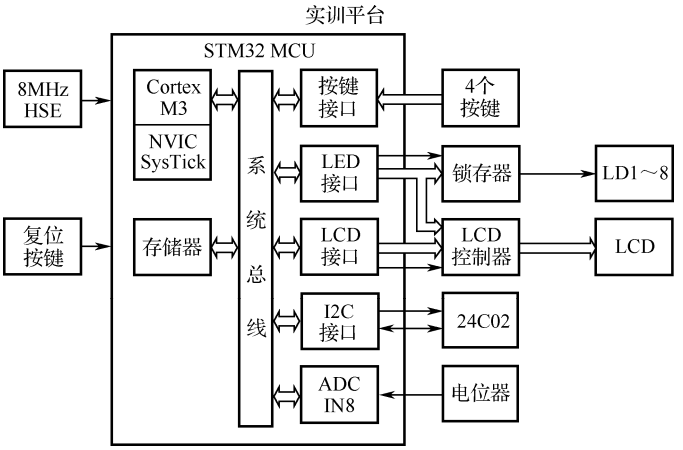



图 6.52　系统详细方框图

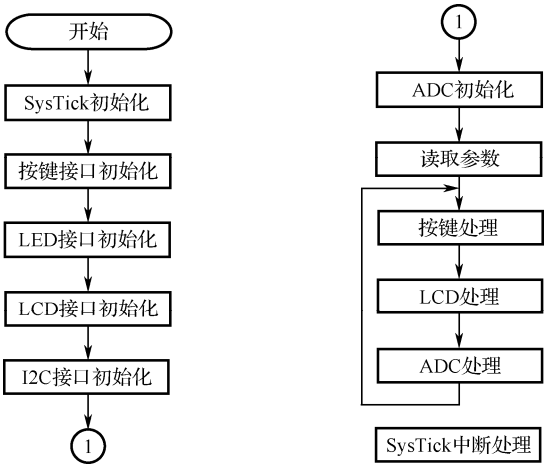



图 6.53　系统主程序流程图

主程序设计如下：

```
#include "key.h"
#include "led.h"
#include "lcd.h"
#include "stdio.h"
#include "i2c.h"
#include "adc.h"

unsigned int  uiVolt;                  // 电压*100
unsigned char ucMax, ucMin;            // 电压*10
unsigned char ucStatus, ucUpper, ucLower;
unsigned char ucState, pucStr[21];
unsigned char ucLed, ucKey_Long;
unsigned long ulTick_ms;
```



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
void KEY_Proc(void);
void LCD_Proc(void);
void ADC_Proc(void);

int main(void)
{
  SysTick_Config(72000);
  KEY_Init();
  LED_Init();

  STM3210B_LCD_Init();
  LCD_Clear(Blue);
  LCD_SetBackColor(Blue);
  LCD_SetTextColor(White);

  i2c_init();
  ADC1_Init();

  i2c_read(pucStr, 0, 4);
  ucMax = pucStr[0];
  ucMin = pucStr[1];
  ucUpper = pucStr[2];
  ucLower = pucStr[3];
  if(ucMax >= 33) ucMax = 24;
  if(ucMin >= 33) ucMin = 12;
  if(ucUpper >= 9)    ucUpper = 1;
  if(ucLower >= 9)    ucLower = 2;

  while(1)
  {
    KEY_Proc();
    LCD_Proc();
    ADC_Proc();
  }
}
```

### 6.7.2 初始化程序设计

初始化程序包括 SysTick 初始化、按键接口初始化、LED 接口初始化、LCD 接口初始化、I2C 接口初始化和 ADC 初始化等。

**1）SysTick 初始化程序设计**

SysTick 初始化用 SysTick_Config()实现。

**2）按键接口初始化程序设计**

按键接口初始化程序使用“key.c”中的 KEY_Init()实现。

**3）LED 接口初始化程序设计**

LED 接口初始化程序使用“led.c”中的 LED_Init()实现。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

4）LCD 接口初始化程序设计

LCD 接口初始化程序使用“lcd.c”中的 STM3210B_LCD_Init()实现。

5）I2C 接口初始化程序设计

I2C 接口初始化程序使用“i2c.c”中的 i2c_init()实现。

6）ADC 初始化程序设计

ADC 初始化程序使用“adc.c”中的 ADC1_Init()实现。

### 6.7.3 处理程序设计

处理程序包括按键处理、LCD 处理、ADC 处理和 LED 处理等，LED 处理在 SysTick 中断处理程序中实现。处理程序流程图如图 6.54 所示。

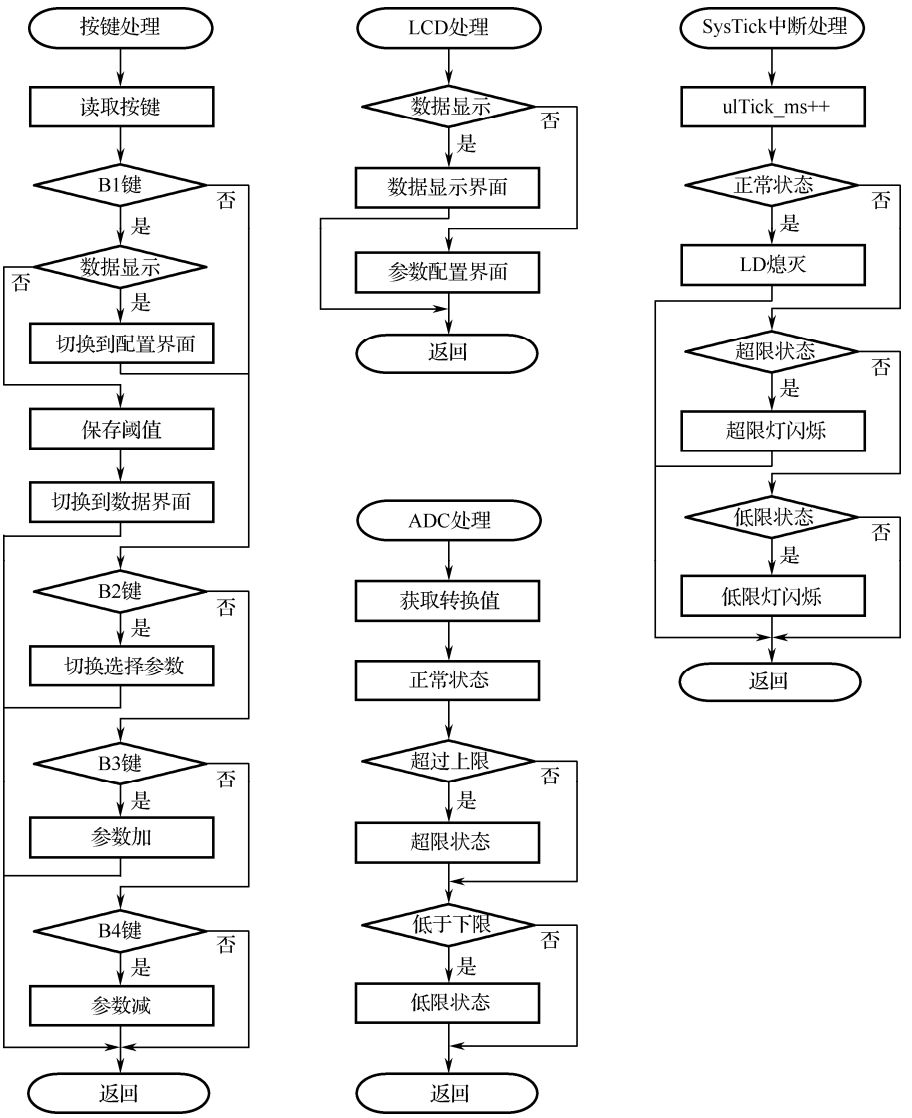



图 6.54　系统处理程序流程图

1）按键处理程序设计

按键处理程序设计如下：



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
void KEY_Proc(void)
{
  unsigned char ucKey_Val;

  ucKey_Val = KEY_Scan();
  if(ucKey_Val != ucKey_Long)
    ucKey_Long = ucKey_Val;
  else
    ucKey_Val = 0;

  switch(ucKey_Val)
  {
    case 1:                              // B1: 设置键
      if(!ucState)
        ucState = 1;
      else
      {
        pucStr[0] = ucMax;
        pucStr[1] = ucMin;
        pucStr[2] = ucUpper;
        pucStr[3] = ucLower;
        i2c_write(pucStr, 0, 4);         // 保存参数
        ucState = 0;
      }
      break;
    case 2:                              // B2: 选择键
      if(ucState)
        if(++ucState == 5)
          ucState = 1;
      break;
    case 3:                              // B3: 参数加
      switch(ucState)
      {
        case 1:
          if(ucMax <= 30) ucMax += 3;
          break;
        case 2:
          if((ucMax - ucMin) > 3) ucMin += 3;
          break;
        case 3:
          if(++ucUpper > 8) ucUpper = 1;
          if(ucUpper == ucLower)
            if(++ucUpper > 8) ucUpper = 1;
          break;
        case 4:
          if(++ucLower > 8) ucLower = 1;
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
        if(ucUpper == ucLower)
          if(++ucLower > 8) ucLower = 1;
      }
      break;
    case 4:                              // B4: 参数减
      switch(ucState)
      {
        case 1:
          if((ucMax - ucMin) > 3) ucMax -= 3;
          break;
        case 2:
          if(ucMin > 3) ucMin -= 3;
          break;
        case 3:
          if(--ucUpper == 0) ucUpper = 8;
          if(ucUpper == ucLower)
          if(--ucUpper == 0) ucUpper = 8;
          break;
        case 4:
          if(--ucLower == 0) ucLower = 8;
          if(ucUpper == ucLower)
          if(--ucLower == 0) ucLower = 8;
      }
  }
}
```

2）LCD 处理程序设计

LCD 处理程序根据 LCD 参考界面设计如下：

```
void LCD_Proc(void)
{
  if(!ucState)
  {
    LCD_DisplayStringLine(Line2, (u8*)"       Main       ");

    sprintf((char*)pucStr, "    Volt: %4.2fV    ", uiVolt/100.0);
    LCD_DisplayStringLine(Line4, pucStr);

    switch(ucStatus)
    {
      case 0:
        LCD_DisplayStringLine(Line5, (u8*)"    Stat: Normal   ");
        break;
      case 1:
        LCD_DisplayStringLine(Line5, (u8*)"    Stat: Upper    ");
        break;
      case 2:
```



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
        LCD_DisplayStringLine(Line5, (u8*)"    Stat: Lower    ");
      }

      sprintf((char*)pucStr, "    %3.1fV  %3.1fV    ",\
        ucMin/10.0, ucMax/10.0);
      LCD_DisplayStringLine(Line6, pucStr);

      sprintf((char*)pucStr, "    LD%1u   LD%1u     ", ucLower, ucUpper);
      LCD_DisplayStringLine(Line7, pucStr);
    }
    else
    {
      LCD_DisplayStringLine(Line2, (u8*)"     Setting");

      if(ucState == 1) LCD_SetBackColor(Red);
      sprintf((char*)pucStr, "   Max Volt: %3.1fV   ", ucMax/10.0);
      LCD_DisplayStringLine(Line4, pucStr);
      LCD_SetBackColor(Blue);

      if(ucState == 2) LCD_SetBackColor(Red);
      sprintf((char*)pucStr, "   Min Volt: %3.1fV   ", ucMin/10.0);
      LCD_DisplayStringLine(Line5, pucStr);
      LCD_SetBackColor(Blue);

      if(ucState == 3) LCD_SetBackColor(Red);
      sprintf((char*)pucStr, "   Upper:    LD%1u    ", ucUpper);
      LCD_DisplayStringLine(Line6, pucStr);
      LCD_SetBackColor(Blue);

      if(ucState == 4) LCD_SetBackColor(Red);
      sprintf((char*)pucStr, "   Lower:    LD%1u    ", ucLower);
      LCD_DisplayStringLine(Line7, pucStr);
      LCD_SetBackColor(Blue);
    }
  }
```

### 3）ADC 处理程序设计

ADC 处理程序设计如下：

```
  void ADC_Proc(void)
  {
    uiVolt = ADC1_Conv()*330/4095;
    ucStatus = 0;
    if(uiVolt > ucMax*10) ucStatus = 1;
    if(uiVolt < ucMin*10) ucStatus = 2;
  }
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

4）LED 处理程序设计

LED 处理用 SysTick 中断处理程序实现，程序设计如下：

```
void SysTick_Handler(void)
{
  ulTick_ms++;

  if(!ucState)
  {
    switch(ucStatus)
    {
      case 0:
        ucLed = 0;
        break;
      case 1:
        if(ulTick_ms%200 == 0) ucLed ^= 1<<(ucUpper-1);
        break;
      case 2:
        if(ulTick_ms%200 == 0) ucLed ^= 1<<(ucLower-1);
    }
  }
  else
    ucLed = 0;
  LED_Disp(ucLed);
}
```

### 6.7.4 系统实现

系统实现的主要步骤如下。

（1）将“206_ADC”文件夹复制并重命名为“307_101”文件夹，在“307_101”文件夹中双击工程文件“STM32.uvproj”打开工程。

（2）修改文件“main.c”的内容。

（3）在 main()后追加 KEY_Proc()、LCD_Proc()和 ADC_Proc()的程序代码。

（4）修改 SysTick_Handler()的程序代码。

### 6.7.5 客观题

（1）STM32 嵌套向量中断控制器具有（　　）个可编程的优先等级。

A．16　　B．32

C．64　　D．42

（2）使用 GPIO 控制 LED 时，要使引脚最大输出速度为 2MHz，应设置相应寄存器的 MODEX[1:0]位值为（　　）。

A．00　　B．01

C．10　　D．11

（3）ADC 转换过程不包含以下哪项？（　　）

A．采样　　B．量化



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

C．编码　　D．逆采样

（4）下列说法中正确的是（　　）。

A．STM32 ADC 是一个 12 位的连续近似模拟到数字的转换器。

B．Cortex M3 可以使用小端格式访问代码。

C．STM32 中，多个外设产生的 DMA 请求，同时只能有一个请求有效。

D．I2C 总线的启动信号和停止信号，只能由主器件发起。

（5）某存储器芯片的存储容量为 8KB，数据线 8 根，地址线为（　　）根。

A．10　　B．11

C．12　　D．13

（6）STM32F103RBT6 每个 DMA 通道具有（　　）事件标志。

A．3　　B．4

C．5　　D．6

（7）放大电路在负载开路时的输出电压为 0.4V，接入 3kΩ的电阻负载后，输出的电压降为 0.3V，则该放大电路的输出电阻为（　　）。

A．10kΩ　　B．2kΩ

C．3kΩ　　D．1kΩ

（8）将三角波转换为矩形波，需选用（　　）。

A．多谐振荡器　　B．双稳态触发器

C．单稳态触发器　　D．施密特触发器

（9）某放大电路中使用的三极管的极限参数为 $P_{CM}=100\text{mW}$，$I_{CM}=20\text{mA}$，$U_{(BR)CEO}=15\text{V}$。以下哪些情况下，三极管不能正常工作？（　　）

A．$U_{CE}=3\text{V}$，$I_C=15\text{mA}$　　B．$U_{CE}=2\text{V}$，$I_C=40\text{mA}$

C．$U_{CE}=6\text{V}$，$I_C=20\text{mA}$　　D．$U_{CE}=9\text{V}$，$I_C=10\text{mA}$

（10）运放电路如图 6.55 所示，输入电压 $U_1=0.4\text{V}$，$U_2=0.8\text{V}$，计算输出电压 $U_o$ 的值为（　　）。

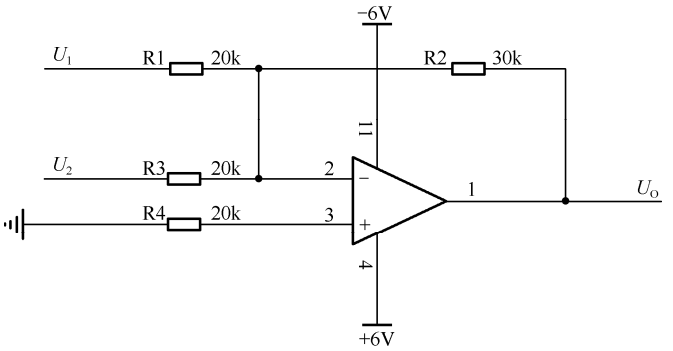



图 6.55　运放电路

A．1.2V　　B．−1.2V

B．−6V　　D．−1.8V

参考答案与评分标准

（1）A（3 分）　（2）C（3 分）　（3）D（3 分）　（4）ABCD（3 分）　（5）D（3 分）

（6）A（3 分）　（7）D（3 分）　（8）D（3 分）　（9）BC（3 分）　（10）D（3 分）

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

## 6.8 第十届决赛试题

系统方框图如图 6.56 所示。

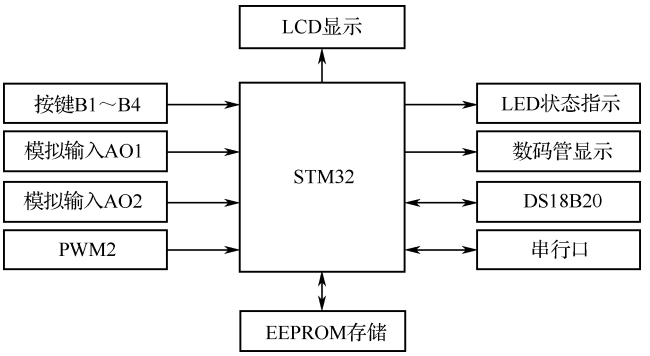



图 6.56　系统方框图

系统功能描述如下：

**1）基本功能**

（1）通过 STM32 内部 ADC 测量资源扩展板电位器 RP5 和 RP6 输出的电压信号。

（2）通过 STM32 内部定时器通道测量资源扩展板上 PWM2 信号占空比。

（3）通过 DS18B20 完成温度测量功能。

（4）通过 EEPROM 完成数据存储功能。

（5）通过按键、LCD 和资源扩展板上数码管完成人机交互功能。

（6）通过 STM32 USART2 完成串口收发功能。

（7）数据测量范围与刷新时间要求。

- 电压数据：0.00～3.30V，刷新时间≤500ms。
- 占空比：PWM2 输出信号占空比，刷新时间≤1s。

**2）LCD 显示**

（1）数据显示界面。

数据显示界面包含 6 个要素：界面名称、RP5 输出电压值 AO1、RP6 输出电压值 AO2、PWM2 输入信号占空比、温度数据和参数变动次数 *N*，数据显示参考界面如图 6.57 所示。

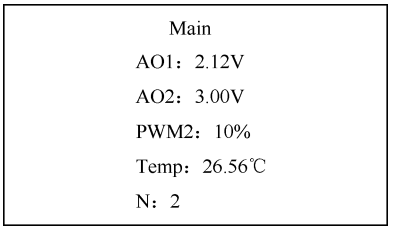



图 6.57　数据显示参考界面

（2）参数配置界面。

参数配置界面包含 3 个要素：界面名称、温度超限参数 *T* 和比较通道参数 *X*，参数配置参考界面如图 6.58 所示。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

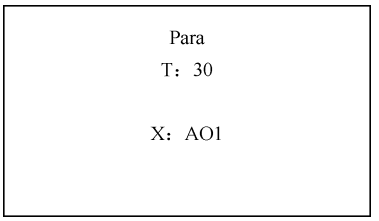



图 6.58　参数配置参考界面

其中，温度设置范围是 20～40℃，比较通道在 AO1 和 AO2 间切换。

**3）数码管功能**

使用资源扩展板上的 3 位数码管，以 2s 为间隔，交替显示当前配置的温度超限参数和比较通道参数。显示格式要求如下。

（1）温度超限参数格式：C30，表示当前温度超限参数为 30℃。

（2）比较通道参数显示格式：AO1 或 AO2。

**4）按键功能**

按键 B1 为设置键：按下时进入参数设置界面，再次按下时返回数据显示界面，并检查参数是否发生变化，如果变化则变动次数加 1，并将次数保存在 EEPROM 中。

按键 B2 为选择键：在参数设置界面下按下选择键可切换设置参数，被选择的参数项高亮显示。

按键 B3 为加键：进行温度超限参数设置时，按下 B3 键，参数加 1℃，长按 B3 键可实现快速加功能；进行比较通道参数设置时，按下 B3 键，切换通道 AO1 或 AO2。

按键 B4 为减键：进行温度超限参数设置时，按下 B3 键，参数减 1℃，长按 B3 键可实现快速减功能；进行比较通道参数设置时，按下 B3 键，切换通道 AO1 或 AO2。

按键 B2、B3 和 B4 仅在设置界面有效，加减操作应进行合理的数据边界保护。

**5）串口通信功能**

（1）通信模式。

- 波特率：9600bps。
- 校验位：无校验。
- 停止位：1 位。
- 通信端口：STM32-USART2。

（2）自动上报。

$V_{IN}>k\times3.3$ 时，以 1s 为间隔自动上报当前的温度数据。

其中，$V_{IN}$ 为 AO1 或 AO2 的输出电压，$k$ 为 PWM2 的占空比。

上报格式为：\$24.32\r\n。

（3）数据召测功能。

通过串口调试软件下发数据召测指令，设备接收到正确指令后，上报数据或参数。召测指令格式要求如下。

- 查询数据指令：ST\r\n，接收到指令后，设备触发一次温度数据上报，上报数据格式同自动上报。
- 查询参数指令：PARA\r\n，接收到指令后，设备返回当前的温度超限参数和比较通道参数，返回数据格式为：#30,AO1\r\n。

**6）LED 显示功能**

（1）串口自动上报条件满足的条件下，指示灯 LD1 点亮，否则熄灭。

（2）当环境温度高于温度参数时，指示灯 LD8 以 0.2s 为间隔闪烁，否则熄灭。

电子工业出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

（3）未涉及的 LED 应处于熄灭状态，不同功能的指示灯状态切换时应互不影响。

**7）参数存储功能**

使用 EEPROM 完成参数变动次数 *N* 的存储，设备重新上电时，应能够从 EEPROM 中载入数据。

**8）初始状态**

（1）设备上电，默认处于数据显示界面。

（2）工作参数不需要保存在 EEPROM 中，每次上电重置为默认值。

- 温度超限参数：30。
- 比较通道参数：AO1。

**9）资源扩展板跳线配置**

资源扩展板跳线配置如图 6.59 所示。

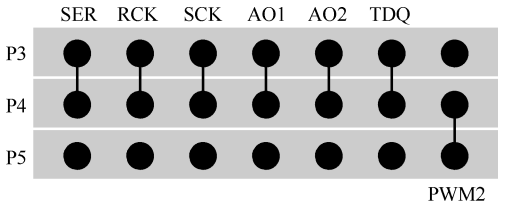



图 6.59　资源扩展板跳线配置

## 6.8.1　系统设计

通过分析系统功能，可以得到系统详细方框图如图 6.60 所示。

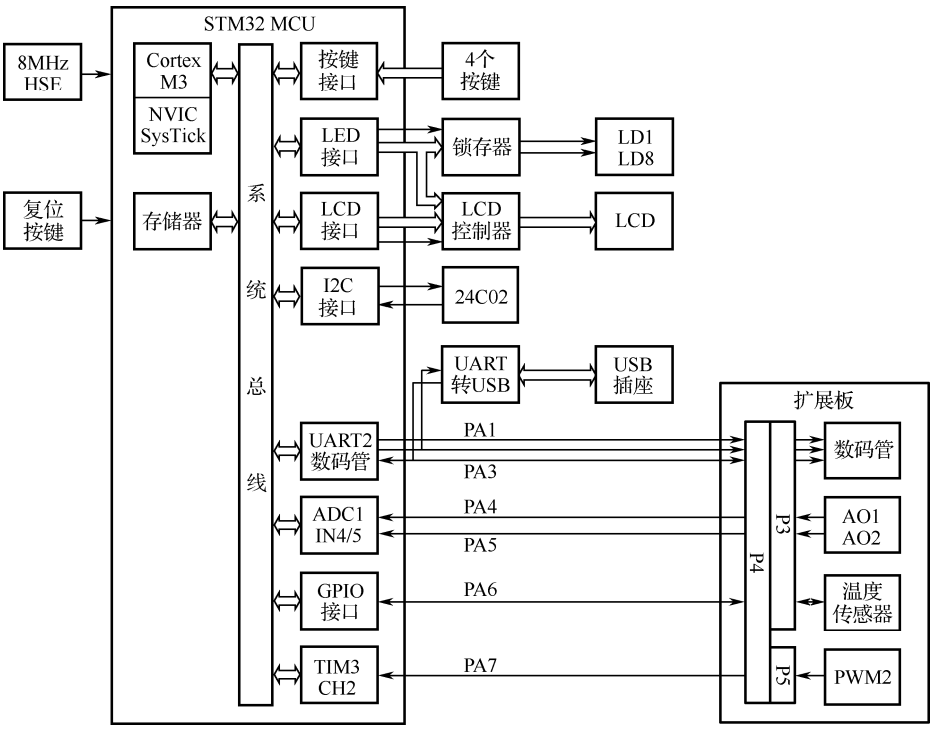



图 6.60　系统详细方框图

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

扩展板上的数码管通过 PA1～PA3 与 MCU 连接，其中 PA2 和 PA3 与 USART2 公用，通过编程分时使用，AO1 和 AO2 分别通过 PA4 和 PA5 与 ADC IN4 和 IN5 连接，温度传感器 DS18B20 通过 PA6 与 MCU 连接，PWM2 通过 PA7 与 TIM3 CH2 连接。

系统主程序流程图如图 6.61 所示。

主程序首先对系统进行初始化，包括 SysTick 初始化、按键接口初始化、LED 接口初始化、LCD 接口初始化、I2C 接口初始化、温度传感器接口初始化、ADC1 初始化和 TIM3 初始化等，数码管接口初始化和 USART2 接口初始化在 SysTick 中断处理程序中实现。系统初始化后从 AT24C02 中读取参数变动次数。

主循环的操作包括按键处理、LCD 处理、UART 处理和数据处理等，按键处理中包含保存参数变动次数，数据处理中包含读取温度值、ADC 转换值和 PWM2 占空比值等，LED 处理和 SEG 处理在 SysTick 中断处理程序中实现，UART 接收在 UART 中断处理程序中实现。

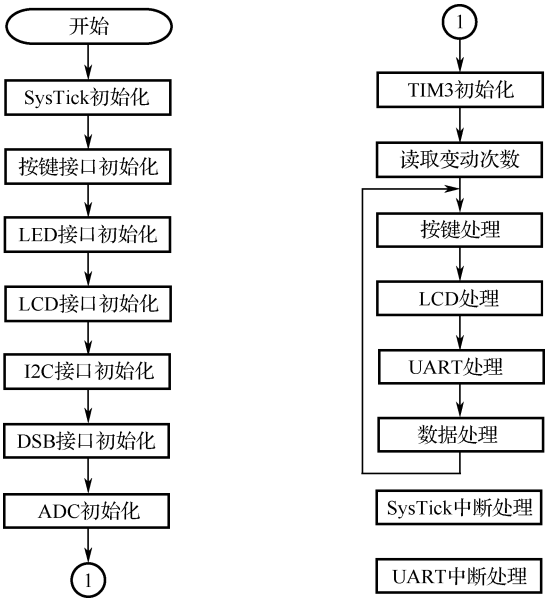



图 6.61　系统主程序流程图

主程序设计如下：

```
#include "key.h"
#include "led.h"
#include "lcd.h"
#include "usart.h"
#include "i2c.h"
#include "adc.h"
#include "tim.h"
#include "seg.h"
#include "ds18b20.h"

#define TIM_CK 1000000

unsigned int  uiAo1, uiAo2, uiTemp;
unsigned char ucPwm, ucNum;
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
unsigned char ucTh0 = 30, ucTh1;
unsigned char ucCh0 = 1, ucCh1;
unsigned char ucState, pucStr[21];
unsigned char pucRcv[6], ucRno;
unsigned char ucSec0, ucSec1;
unsigned char ucLed, ucKey_Long;
unsigned long ulTick_ms, ulKey_Time;

void KEY_Proc(void);
void LCD_Proc(void);
void UART_Proc(void);
void DATA_Proc(void);

int main(void)
{
  SysTick_Config(72000);
  KEY_Init();
  LED_Init();

  STM3210B_LCD_Init();
  LCD_Clear(Blue);
  LCD_SetBackColor(Blue);
  LCD_SetTextColor(White);

  i2c_init();
  ds18b20_init_x();
  ADC1_Init_AKEY();
  TIM3_ICInit_102();

  i2c_read(pucStr, 0, 1);
  ucNum = pucStr[0];

  while(1)
  {
    KEY_Proc();
    LCD_Proc();
    UART_Proc();
    DATA_Proc();
  }
}
```

### 6.8.2 初始化程序设计

初始化程序包括 SysTick 初始化、按键接口初始化、LED 接口初始化、LCD 接口初始化、I2C 接口初始化、温度传感器接口初始化、ADC1 初始化和 TIM3 初始化等。



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

1）SysTick 初始化程序设计

SysTick 初始化用 SysTick_Config()实现。

2）按键接口初始化程序设计

按键接口初始化程序使用“key.c”中的 KEY_Init()实现。

3）LED 接口初始化程序设计

LED 接口初始化程序使用“led.c”中的 LED_Init()实现。

4）LCD 接口初始化程序设计

LCD 接口初始化程序使用“lcd.c”中的 STM3210B_LCD_Init()实现。

5）I2C 接口初始化程序设计

I2C 接口初始化程序使用“i2c.c”中的 i2c_init()实现。

6）温度传感器接口初始化程序设计

温度传感器接口初始化程序使用“ds18b20.c”中的 ds18b20_init_x()实现。

7）ADC 初始化程序设计

ADC 初始化程序在“adc.c”中 ADC1_Init_AKEY()的基础上修改实现，内容如下：

```
void ADC1_Init_AKEY(void)
{
  GPIO_InitTypeDef GPIO_InitStruct;
  ADC_InitTypeDef ADC_InitStruct;

  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1, ENABLE);

  GPIO_InitStruct.GPIO_Pin = GPIO_Pin_4 | GPIO_Pin_5;
  GPIO_InitStruct.GPIO_Mode = GPIO_Mode_AIN;
  GPIO_Init(GPIOA, &GPIO_InitStruct);

  ADC_InitStruct.ADC_Mode = ADC_Mode_Independent;
  ADC_InitStruct.ADC_ScanConvMode = DISABLE;
  ADC_InitStruct.ADC_ContinuousConvMode = DISABLE;
  ADC_InitStruct.ADC_ExternalTrigConv = ADC_ExternalTrigConv_None;
  ADC_InitStruct.ADC_DataAlign = ADC_DataAlign_Right;
  ADC_InitStruct.ADC_NbrOfChannel = 1;
  ADC_Init(ADC1, &ADC_InitStruct);

  ADC_RegularChannelConfig(ADC1, ADC_Channel_5, 1,
    ADC_SampleTime_1Cycles5);
  ADC_InjectedChannelConfig(ADC1, ADC_Channel_4, 1,
    ADC_SampleTime_1Cycles5);
  ADC_AutoInjectedConvCmd(ADC1, ENABLE);

  ADC_Cmd(ADC1, ENABLE);
  ADC_StartCalibration(ADC1);
  while(ADC_GetCalibrationStatus(ADC1));
}
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

8）TIM3 初始化程序设计

TIM3 初始化程序在“tim.c”中 TIM2_ICInit()的基础上修改实现，内容如下：

```
void TIM3_ICInit_102(void)
{
  GPIO_InitTypeDef GPIO_InitStruct;
  TIM_TimeBaseInitTypeDef TIM_TimeBaseInitStruct;
  TIM_ICInitTypeDef  TIM_ICInitStruct;

  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
  RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3, ENABLE);

  GPIO_InitStruct.GPIO_Pin = GPIO_Pin_7;
  GPIO_InitStruct.GPIO_Mode = GPIO_Mode_IN_FLOATING;
  GPIO_Init(GPIOA, &GPIO_InitStruct);

  TIM_TimeBaseInitStruct.TIM_Prescaler = 71;
  TIM_TimeBaseInitStruct.TIM_CounterMode = TIM_CounterMode_Up;
  TIM_TimeBaseInitStruct.TIM_Period = 65535;
  TIM_TimeBaseInitStruct.TIM_ClockDivision = TIM_CKD_DIV1;
  TIM_TimeBaseInit(TIM3, &TIM_TimeBaseInitStruct);

  TIM_ICInitStruct.TIM_Channel = TIM_Channel_2;
  TIM_ICInitStruct.TIM_ICPolarity = TIM_ICPolarity_Rising;
  TIM_ICInitStruct.TIM_ICSelection = TIM_ICSelection_DirectTI;
  TIM_ICInitStruct.TIM_ICPrescaler = TIM_ICPSC_DIV1;
  TIM_ICInitStruct.TIM_ICFilter = 0;
  TIM_PWMICInit(TIM3, &TIM_ICInitStruct);

  TIM_SelectInputTrigger(TIM3, TIM_TS_TI2FP2);
  TIM_SelectSlaveMode(TIM3, TIM_SlaveMode_Reset);
  TIM_Cmd(TIM3, ENABLE);
}

unsigned char TIM3_Pwm(void)
{
  if(TIM_GetFlagStatus(TIM3, TIM_FLAG_CC2))
    return ((TIM_GetCapture1(TIM3)+1)*100/(TIM_GetCapture2(TIM3)+1));
  else
    return 0;
}
```

## 6.8.3 处理程序设计

处理程序包括按键处理、LCD 处理、UART 处理和数据处理等，按键处理中包含保存参数变动次数，数据处理中包含读取温度值、ADC 转换值和 PWM2 占空比值等，LED 处理和 SEG 处理



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

在 SysTick 中断处理程序中实现，UART 接收在 UART 中断处理程序中实现。按键处理、LCD 处理、UART 处理、数据处理和 SysTick 中断处理程序流程图如图 6.62 所示。

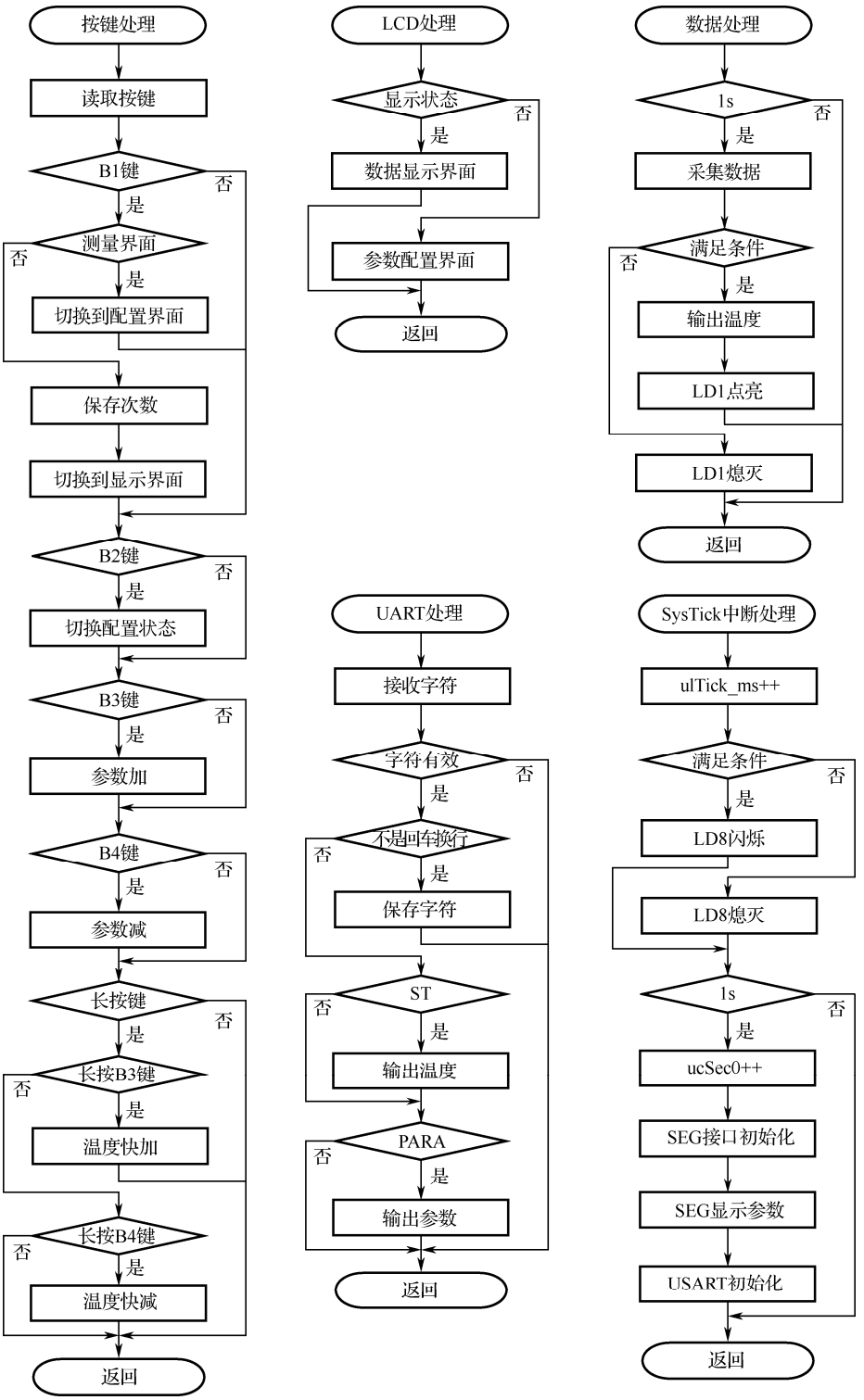



图 6.62　系统处理程序流程图

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

1）按键处理程序设计

按键处理程序设计如下：

```
void KEY_Proc(void)
{
  unsigned char ucKey_Val;

  ucKey_Val = KEY_Scan();
  if(ucKey_Val != ucKey_Long)
  {
    ucKey_Long = ucKey_Val;
    ulKey_Time = ulTick_ms;
  }
  else
    ucKey_Val = 0;

  switch(ucKey_Val)
  {
    case 1:                              // B1: 设置键
      if(!ucState)
      {
        ucTh1 = ucTh0;                   // 暂存设置值
        ucCh1 = ucCh0;
        ucState = 1;                     // 进入设置状态
      }
      else
      {
        if((ucTh1 != ucTh0) || (ucCh1 != ucCh0))
        {                                // 设置值变化
          pucStr[0] = ++ucNum;
          i2c_write(pucStr, 0, 1);       // 保存次数
        }
        ucState = 0;                     // 进入显示状态
      }
      break;
    case 2:                              // B2: 选择键
      if(ucState)
        if(++ucState == 3)
          ucState = 1;
      break;
    case 3:                              // B3: 加键
      if(ucState == 1)
        if(ucTh0 < 40) ucTh0++;
      if(ucState == 2)
      {
        if(ucCh0 == 1) ucCh0 = 2;
```



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
        else ucCh0 = 1;
      }
      break;
    case 4:                          // B4: 减键
      if(ucState == 1)
        if(ucTh0 > 20) ucTh0--;
      if(ucState == 2)
      {
        if(ucCh0 == 1) ucCh0 = 2;
        else ucCh0 = 1;
      }
  }
  if(ulTick_ms-ulKey_Time > 800)
  {
    switch(ucKey_Long)
    {
      case 3:                        // B3: 快速加
        if(ucState == 1)
          if(ucTh0 < 40) ucTh0++;
        break;
      case 4:                        // B4: 快速减
        if(ucState == 1)
          if(ucTh0 > 20) ucTh0--;
    }
  }
}
```

### 2）LCD 处理程序设计

LCD 处理程序根据 LCD 参考界面设计如下：

```
void LCD_Proc(void)
{
  if(!ucState)                       // 显示状态
  {
    LCD_DisplayStringLine(Line1, (u8*)"       Main       ");

    sprintf((char*)pucStr, "    AO1: %4.2fV    ", uiAo1*3.3/4095);
    LCD_DisplayStringLine(Line3, pucStr);

    sprintf((char*)pucStr, "    AO2: %4.2fV    ", uiAo2*3.3/4095);
    LCD_DisplayStringLine(Line4, pucStr);

    sprintf((char*)pucStr, "    PWM: %2d%%      ", ucPwm);
    LCD_DisplayStringLine(Line5, pucStr);

    sprintf((char*)pucStr, "    Temp:%5.2fC    ", uiTemp/16.0);
    LCD_DisplayStringLine(Line6, pucStr);
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
    sprintf((char*)pucStr, "    Num: %05d    ", ucNum);
    LCD_DisplayStringLine(Line7, pucStr);
  }
  else                                  // 设置状态
  {
    LCD_DisplayStringLine(Line1, (u8*)"       Para       ");

    sprintf((char*)pucStr, "      T: %2dC      ", ucTh0);
    if(ucState == 1) LCD_SetBackColor(Red);
    LCD_DisplayStringLine(Line3, pucStr);
    LCD_SetBackColor(Blue);

    sprintf((char*)pucStr, "      X: AO%1d      ", ucCh0);
    if(ucState == 2) LCD_SetBackColor(Red);
    LCD_DisplayStringLine(Line5, pucStr);
    LCD_SetBackColor(Blue);

    LCD_DisplayStringLine(Line4, (u8*)"                ");
    LCD_DisplayStringLine(Line6, (u8*)"                ");
    LCD_DisplayStringLine(Line7, (u8*)"                ");
  }
}
```

### 3）UART 处理程序设计

UART 处理程序设计如下：

```
void UART_Proc(void)
{
  if(ucRno)
  {
    if(pucRcv[ucRno-1] == 0xa)
    {
      if((pucRcv[0]=='S') && (pucRcv[1]=='T') && (pucRcv[2]==0xd))
        printf("$%5.2f\r\n", uiTemp/16.0);
      else if((pucRcv[0]=='P') && (pucRcv[1]=='A')\
        && (pucRcv[2]=='R') && (pucRcv[3]=='A') && (pucRcv[4]==0xd))
        printf("#%2d,AO%1d\r\n", ucTh0, ucCh0);
      else
        printf("ERROR\r\n");
      ucRno = 0;
    }
    else if(ucRno == 6)
    {
      printf("ERROR\r\n");
      ucRno = 0;
    }
```



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
  }
}
```

4）数据处理程序设计

数据处理程序设计如下：

```
void DATA_Proc(void)
{
  if(ucSec1 != ucSec0)
  {
    ucSec1 = ucSec0;

    uiAo1 = ADC1_InjeConv();
    uiAo2 = ADC1_Conv();
    uiTemp = dsb_read();
    ucPwm = TIM3_Pwm();

    if((uiAo1 > ucPwm*41) || (uiAo2 > ucPwm*41))
    {
      printf("$%5.2f\r\n", uiTemp/16.0);
      ucLed |= 1;
    }
    else
      ucLed &= 0xfe;
  }
}
```

5）SysTick 中断处理程序设计

SysTick 中断处理程序设计如下：

```
void SysTick_Handler(void)
{
  ulTick_ms++;

  if((uiTemp>>4) >= ucTh0)
  {
    if(!(ulTick_ms % 200)) ucLed ^= 0x80;
  }
  else
    ucLed &= 0x7f;
  LED_Disp(ucLed);

  if(ulTick_ms == 1000)
  {
    ulTick_ms = 0;
    ucSec0++;

    SEG_Init();
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
    if(ucSec0 & 2)
      SEG_Disp(0xc, ucTh0/10, ucTh0%10, 0);
    else
      SEG_Disp(0xa, 0, ucCh0, 0);
    USART2_Init(9600);
  }
}
```

6）UART 中断处理程序设计

UART 中断处理程序设计如下：

```
void USART2_IRQHandler(void)
{
  pucRcv[ucRno++] = USART_ReceiveData(USART2);
}
```

### 6.8.4 系统实现

系统实现的主要步骤如下：

（1）将“207_TIM”文件夹复制并重命名为“308_102”文件夹，在“308_102”文件夹中双击工程文件“STM32.uvproj”打开工程。

（2）修改“adc.c”中 ADC1_Init_AKEY()的内容。

（3）在“tim.h”中添加下列函数声明：

```
void TIM3_ICInit_102(void);
unsigned char TIM3_Pwm(void);
```

（4）在“tim.c”中将 TIM2_ICInit()函数代码复制并重命名为 TIM3_ICInit_102()函数代码。

（5）修改 TIM3_ICInit_102()函数代码。

（6）在“tim.c”中追加 TIM3_Pwm()函数代码。

（7）将“seg.c”和“ds18b20.c”添加到“Driver”组。

（8）修改文件“main.c”的内容。

（9）在 main()后追加 KEY_Proc()、LCD_Proc()、UART_Proc()和 DATA_Proc()的程序代码。

（10）修改 SysTick_Handler()的程序代码。

（11）添加 USART_IRQHandler()的程序代码。

### 6.8.5 客观题

（1）STM32 启动模式包括（　　）。

A．主闪存存储器启动　　B．系统存储器启动

C．内置 SRAM 存储器启动　　D．BKP 备份区域启动

（2）RLC 串联电路的谐振频率为 $F_0$ = 1000Hz，当频率为 800Hz 的正弦电压源激励时该电路呈（　　）。

A．容性　　B．感性

C．阻性　　D．不能确定

（3）下列有关信号的说法中错误的是（　　）。

A．信号是消息的表现形式　　B．信号都可以用一个确定的时间函数来描述



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

C．声音和图像都是信号　　D．信号可以分解为周期信号和非周期信号

（4）由 5 个 D 触发器构成的环形计数器，其计数长度为（　　）。

A．5　　B．10

C．25　　D．32

（5）在数字电路中，三极管相当于一个开关，通常工作在（ ）状态。

A．放大　　B．饱和

C．截止　　D．死区

（6）以下关于 USB 通信的说法中错误的是（　　）。

A．串行通信方式　　B．不可热插拔

C．通信速率比 RS-232 快　　D．通过一对差分信号线传输数据

（7）以下哪项为非文件系统格式？（　　）

A．FAT　　B．NTFS

C．Ext　　D．DOS

（8）Cortex M3 处理器中的寄存器 r14 代表（　　）。

A．通用寄存器　　B．链接寄存器

C．程序计数器　　D．程序状态寄存器

（9）由理想二极管组成的电路如图 6.63 所示，AB 两端的电压 $U_{AB}$ 为（　　）。

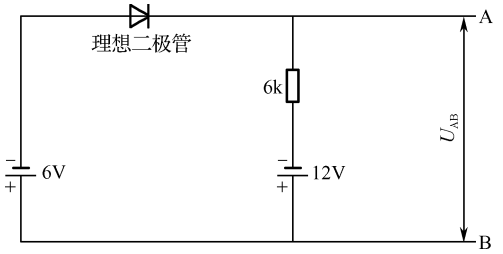



图 6.63　理想二极管电路

A．18V　　B．+12V

C．−6V　　D．−12V

（10）两个逻辑函数恒等，则它们必然具有唯一的（　　）。

A．逻辑表达式　　B．真值表

C．电路图　　D．逻辑图形符号

（11）以下关于差分信号的说法中正确的是（　　）。

A．差分信号在 PCB 布线处理中，一般要求等间距、等长处理

B．CAN、RS-485、RS-232 均为差分接收方式，具有较好的抑制共模干扰能力

C．采用差分信号进行数据传输的通信方式中，数据接收端与发送端必须共地

D．数据接收端通过比较驱动端发送的两个电压信号差值来判断逻辑状态

（12）NAND FLASH 存储器和 NOR FLASH 存储器的区别是（　　）。

A．NOR FLASH 读速度较慢　　B．NAND FLASH 写速度较慢

C．NAND FLASH 擦除速度较慢　　D．NAND FLASH 擦除单元较小

（13）图 6.64 所示的运算放大器电路中，$U_I$ = 2V，运放通过 DC 12V 单电源供电，输出端电压 $U_O$ 为（　　）。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

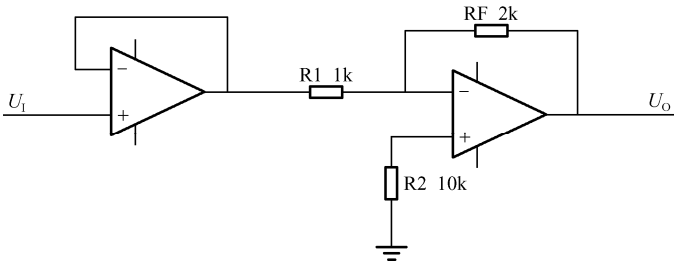



图 6.64　运算放大器电路

A．4V　　B．1V

C．−4V　　D．以上均不正确

（14）不具有压电效应的滤波器是（　　）。

A．石英晶体滤波器　　B．LC 滤波器

C．RC 滤波器　　D．声表面波滤波器

（15）以下关于 ARM Cortex M3 处理器的说法中错误的是（　　）。

A．Cortex-M3 系列处理器不支持 Thumb-2 指令集

B．STM32 IO 端口寄存器必须以 32 位字的方式访问

C．Cortex-M3 处理器可以使用 4 个堆栈

D．中断控制器允许存在相同的优先级

参考答案与评分标准

（1）ABC（2分）　（2）A（2分）　（3）B（2分）　（4）A（2分）　（5）BC（2分）

（6）B（2分）　（7）D（2分）　（8）B（2分）　（9）C（2分）　（10）B（2分）

（11）AD（2分）　（12）D（2分）　（13）D（2分）　（14）BC（2分）　（15）AC（2分）



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

# 附录 A　国信长天单片机竞赛实训平台

国信长天单片机竞赛实训平台包括下列功能模块：

（1）单片机芯片

- 配置 IAP15F2K61S2 单片机

（2）显示模块

- 配置 8 路 LED 输出 L1～L8
- 配置 8 位 8 段共阳极数码管 DS1～DS2
- 配置 LCM 1602 和 12860 液晶接口

（3）输入/输出模块

- 配置 4×4 矩阵按键 S4～S19，其中 S4～S7 可通过跳线配置为独立按键
- 配置继电器和蜂鸣器
- 配置功率放大电路驱动扬声器

（4）传感模块

- 配置光敏电阻
- 配置数字温度传感器 DS18B20
- 配置红外发射管及红外一体头 1838
- 配置超声波收发探头及相应的驱动电路

（5）电源

- USB 和外接 5V 直流电源双电源供电

（6）通信功能

- 板载 USB 转串口功能，可以完成单片机与 PC 的串行通信
- 单总线扩展，可以外接其他单总线接口器件
- I2C 总线，可以做 I2C 总线实验

（7）存储/IO 扩展

- 配置 EEPROM 芯片 AT24C02

（8）程序下载

- 板载 USB 下载功能，不需要另外配备编程器
- 板载 USB 转串口功能，可以对支持串行下载功能的芯片进行程序下载

（9）ADC/DAC 模块

- 配置 PCF8591 ADC/DAC 芯片，内含 8 位 4 通道 ADC 和单通道 DAC

（10）信号发生模块

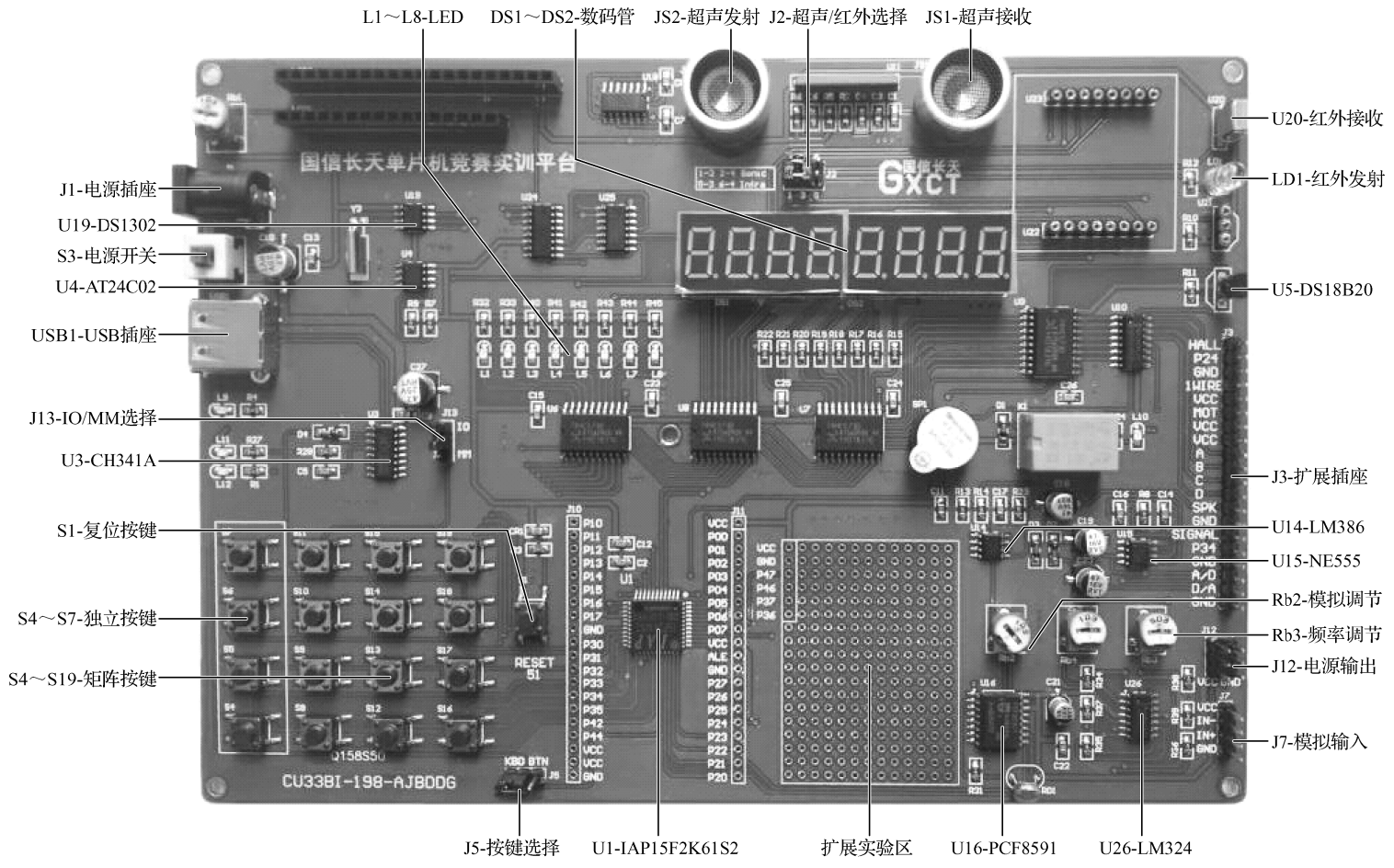
- 配置 555 方波发生器，可以产生实验所需的 200Hz～20kHz 的方波信号

（11）其他

- 配置信号放大模块，可以对输入的低电压模拟信号进行放大
- 外设可以用存储器映射方式访问，也可以直接控制 I/O 口访问

国信长天单片机竞赛实训平台实物图如图 A.1 所示。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY




图A.1　国信长天单片机竞赛实训平台实物图

电子工业出版社 PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

国信长天单片机竞赛实训平台简化方框图如图 A.2 所示。

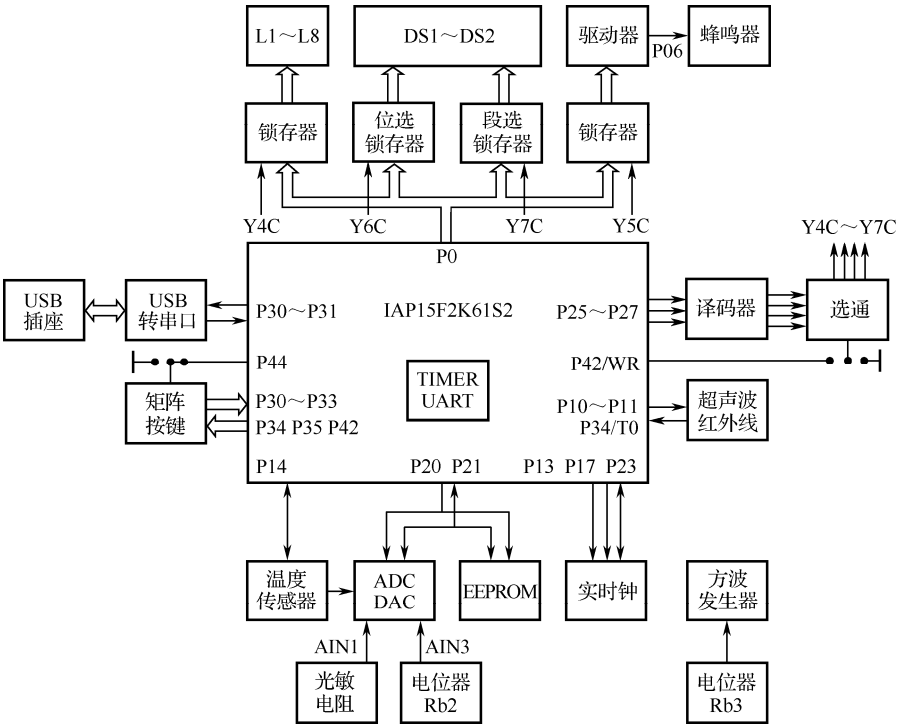



图 A.2　国信长天单片机竞赛实训平台简化方框图

国信长天单片机竞赛实训平台设备连接关系如表 A.1 所示。

**表 A.1　国信长天单片机竞赛实训平台设备连接关系**

| MCU 引脚 | 连　接 | 设　备 | 名　称 | 功 能 说 明 |
|---|---|---|---|---|
| P30～P33 | 行线 | 矩阵<br>按键 | S4～S19 | 用户按键 S4～S19 |
| P34/P35/P42/P44 | 列线 | | | |
| P00～P08 | 数据 | | | LED、数码管和驱动器公用 |
| P25～P27 | Y4C | LED | L1～L8 | LED 锁存器使能 |
| | Y5C | 驱动器 | UNL2003 | 驱动器锁存器使能 |
| | Y6C | 数码管 | DS1～DS2 | 数码管位选锁存器使能 |
| | Y7C | | | 数码管段选锁存器使能 |
| P10～P11 | J2.1-J2.3/J2.2-J2.4 | 超声波 | | 超声波发送/接收 |
| | J2.3-J2.5/J2.4-J2.6 | 紫外线 | | 红外线发送/接收 |
| P13/P17/P23 | RST/SCK/IO | 实时钟 | DS1302 | |
| P14 | DQ | 温度传感器 | DS13B20 | |
| P20/P21 | SCL/SDA | EEPROM | AT24C02 | |
| | | ADC/DAC | PCF8591 | CH1—光敏，CH3—Rb2 |
| P34 | J3.15-J3.16 | 方波发生器 | NE555N | Rb3 |

国信长天单片机竞赛实训平台电路图如图 A.3 所示。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

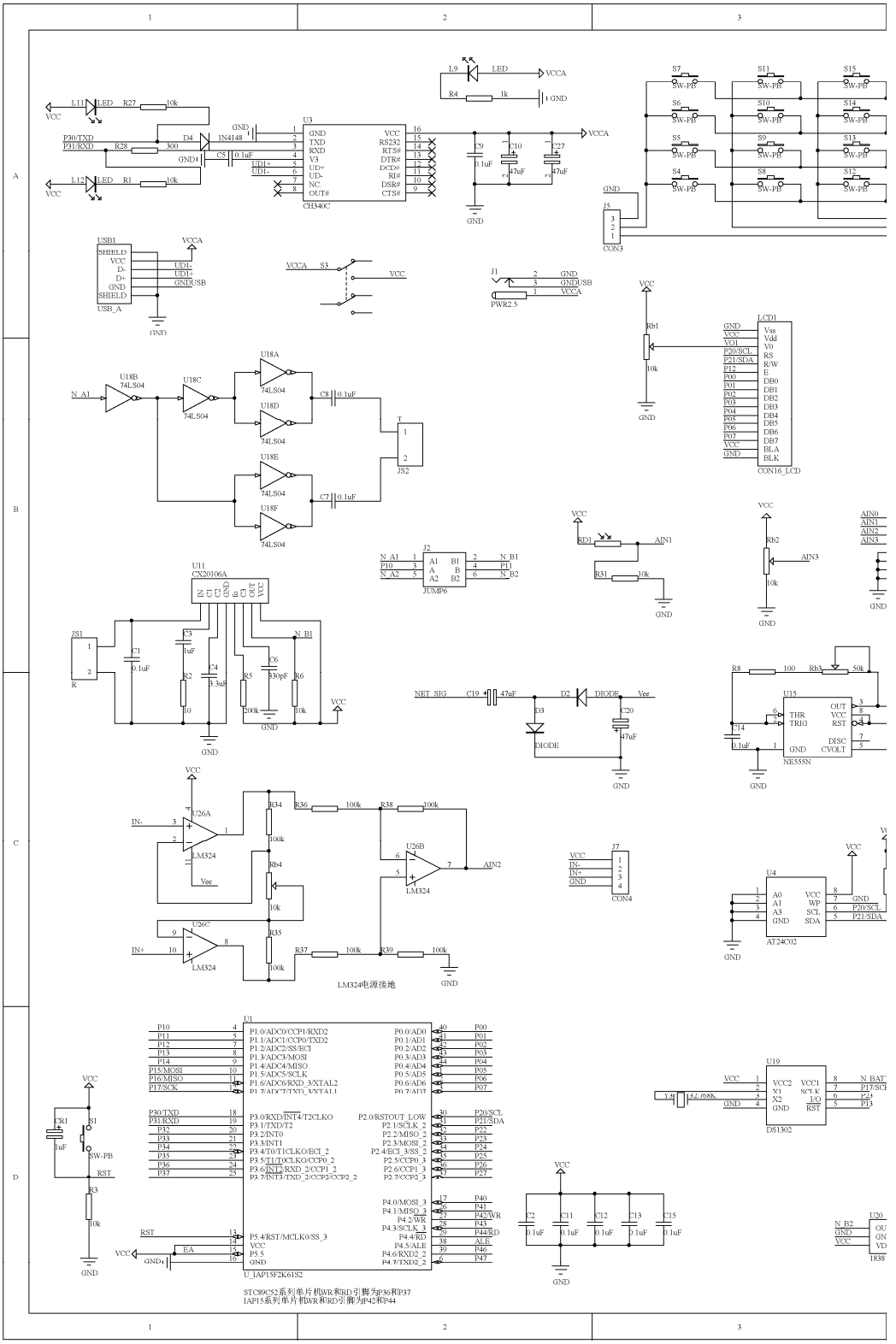
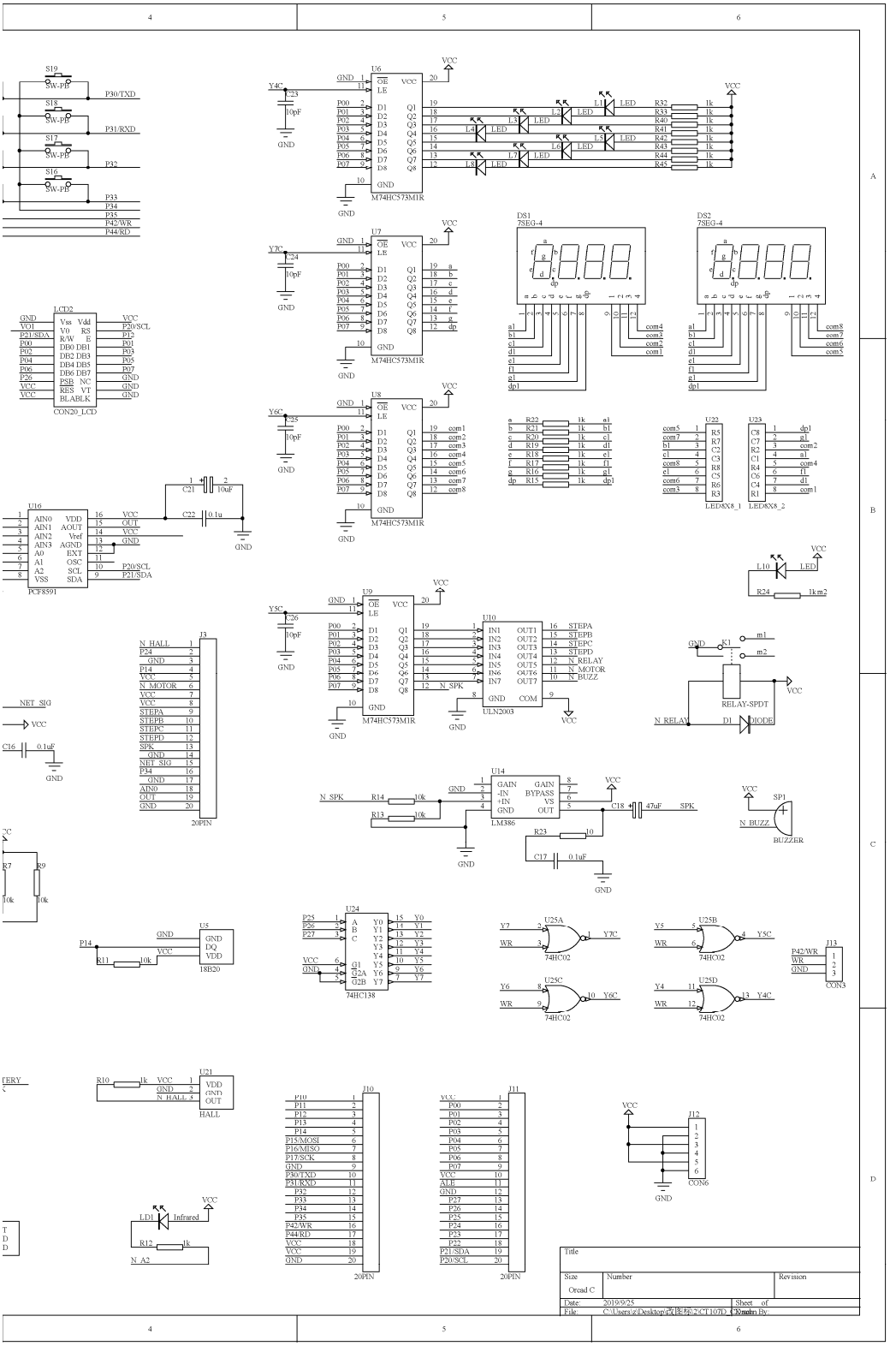



图 A.3　国信长天单片机

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY




竞赛实训平台电路图

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

# 附录 B　国信长天嵌入式竞赛实训平台

国信长天嵌入式竞赛实训平台由以下功能模块组成：

- MCU：STM32F103RBT6（U2）
- 4 个用户按键（B1～B4）、1 个有源蜂鸣器（LS1）
- 8 个用户 LED（LD1～LD8）、2.4 寸 TFT-LCD
- 2 个 UART 接口（UART1 使用电平转换，UART2 使用 UART-USB 转换）
- 1 个 SD 卡插座（CN4）、1 个 EEPROM 芯片 24C02（U6）
- 1 个可调模拟电压输入（R37）
- 1 个扩展插座（J3）、1 个 USB 设备接口（CN5）
- 板载 Coocox 调试器（USB 接口 CN2，无需外接调试器）

国信长天嵌入式竞赛实训平台方框图如图 B.1 所示。

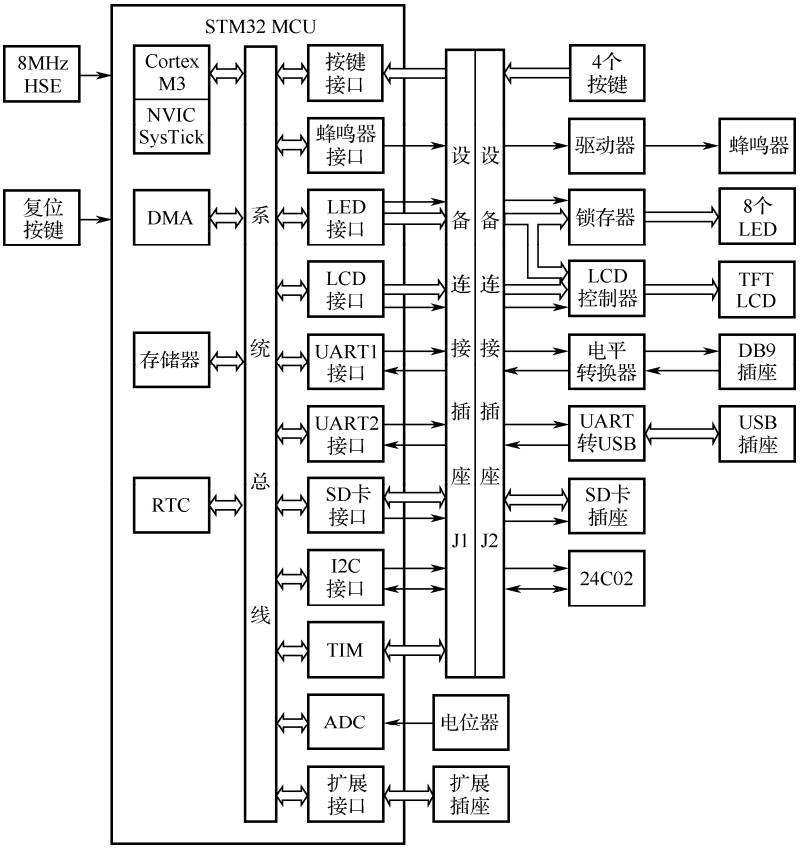



图 B.1　国信长天嵌入式竞赛实训平台方框图

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

国信长天嵌入式竞赛实训平台设备连接关系如表 B.1 所示。

表 B.1　国信长天嵌入式竞赛实训平台设备连接关系

| 设　备 | 名　称 | 连　接 | MCU 引脚 | 功 能 说 明 |
|---|---|---|---|---|
| 按键 | B1 | J1.17-J2.17 | PA0 | 用户按键 1 |
| | B2 | J1.18-J2.18 | PA8 | 用户按键 2 |
| | B3 | J1.19-J2.19 | PB1 | 用户按键 3 |
| | B4 | J1.20-J2.20 | PB2 | 用户按键 4 |
| | B5 | – | NRST | 复位按键 |
| 蜂鸣器 | LS1 | J1.16-J2.16 | PB4 | 蜂鸣器（通过三极管驱动） |
| LED | LE | J1.15-J2.15 | PD2 | 用户 LED 数据锁存器使能 |
| | LD1～LD8 | J1.33～40-J2.33～40 | PC8～PC15 | 用户 LED（数据通过 U7 锁存） |
| | LD9 | – | – | 电源指示 |
| LCD（CN3） | CS# | J1.21-J2.21 | PB9 | LCD 片选 |
| | RS | J1.22-J2.22 | PB8 | LCD 寄存器选择 |
| | WR# | J1.23-J2.23 | PB5 | LCD 写选通 |
| | RD# | J1.24-J2.24 | PB10 | LCD 读选通（与 SD 卡 SD1 公用） |
| | PD1～PD8 | J1.25～32-J2.25～32 | PC0～PC7 | LCD 数据低 8 位 |
| | PD10～PD17 | J1.33～40-J2.33～40 | PC8～PC15 | LCD 数据高 8 位 |
| UART1（CN6） | RXD1 | J1.5-J2.5 | PA10 | UART1_RXD（数据通过 U5 转换） |
| | TXD1 | J1.6-J2.6 | PA9 | UART1_TXD（数据通过 U5 转换） |
| UART2（CN2） | RXD2 | J1.3-J2.3 | PA3 | UART2_RXD（数据通过 U3 转换） |
| | TXD2 | J1.4-J2.4 | PA2 | UART2_TXD（数据通过 U3 转换） |
| SD 卡 SPI2（CN4） | SD0 | J1.7-J2.7 | PB14 | SD 卡数据 0（SPI2_MISO） |
| | SD1 | J1.8-J2.8 | PB10 | SD 卡数据 1（LCD 读选通公用） |
| | SD2 | J1.9-J2.9 | PB11 | SD 卡数据 2 |
| | SD3 | J1.10-J2.10 | PB12 | SD 卡数据 3（SPI2_NSS） |
| | CMD | J1.11-J2.11 | PB15 | SD 卡命令（SPI2_MOSI） |
| | CLK | J1.12-J2.12 | PB13 | SD 卡时钟（SPI2_SCK） |
| 24C02（U6） | SCL | J1.13-J2.13 | PB6 | I2C1_SCL |
| | SDA | J1.14-J2.14 | PB7 | I2C1_SDA |

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

续表

| 设　备 | 名　称 | 连　接 | MCU 引脚 | 功 能 说 明 |
|---|---|---|---|---|
| 电位器 | R37 | – | PB0 | ADC_IN8 |
| 扩展插座（J3） | – | J3.4 | PA1 | ADC_IN1/TIM2_CH2 |
| | – | J3.5 | PA2 | ADC_IN2/TIM2_CH3 |
| | – | J3.6 | PA3 | ADC_IN3/TIM2_CH4 |
| | – | J3.7 | PA4 | ADC_IN4/SPI1_NSS |
| | – | J3.8 | PA5 | ADC_IN5/SPI1_SCK |
| | – | J3.9 | PA6 | ADC_IN6/SPI1_MISO/TIM3_CH1 |
| | – | J3.10 | PA7 | ADC_IN7/SPI1_MOSI/TIM3_CH2 |
| 电源选择（JP1） | USB | 1-2 | – | USB 供电（CN2） |
| | Ext | 2-3 | – | 外部供电（CN1） |

国信长天嵌入式竞赛实训平台实物图如图 B.2 和图 B.3 所示。

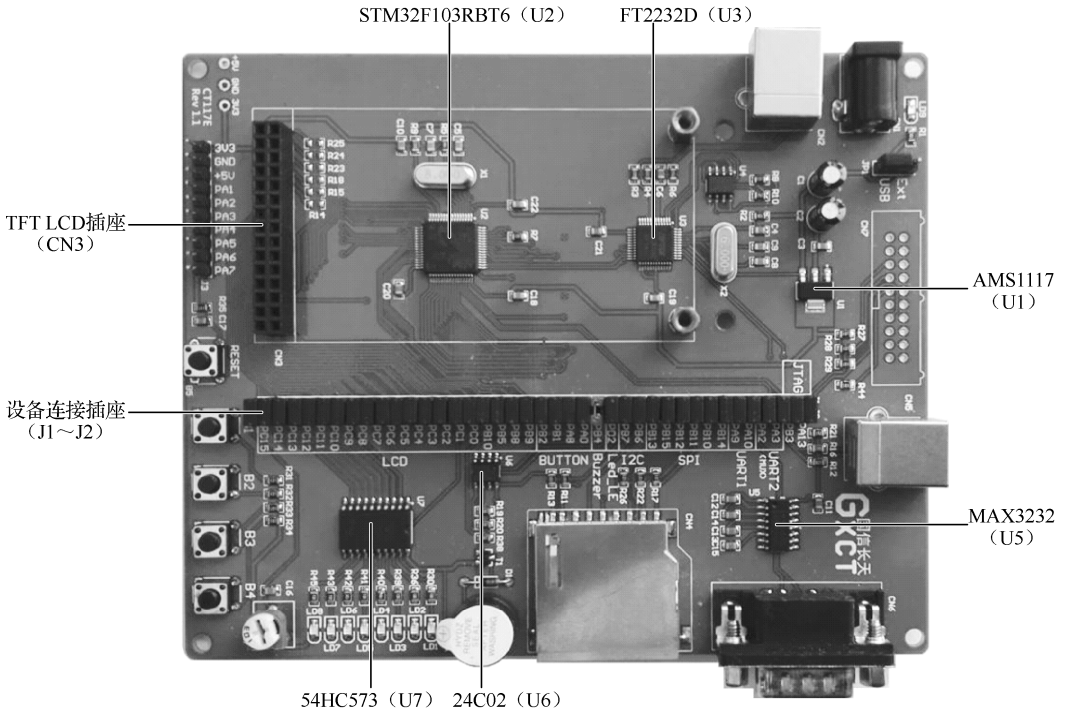



图 B.2　国信长天嵌入式竞赛实训平台实物图 1



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

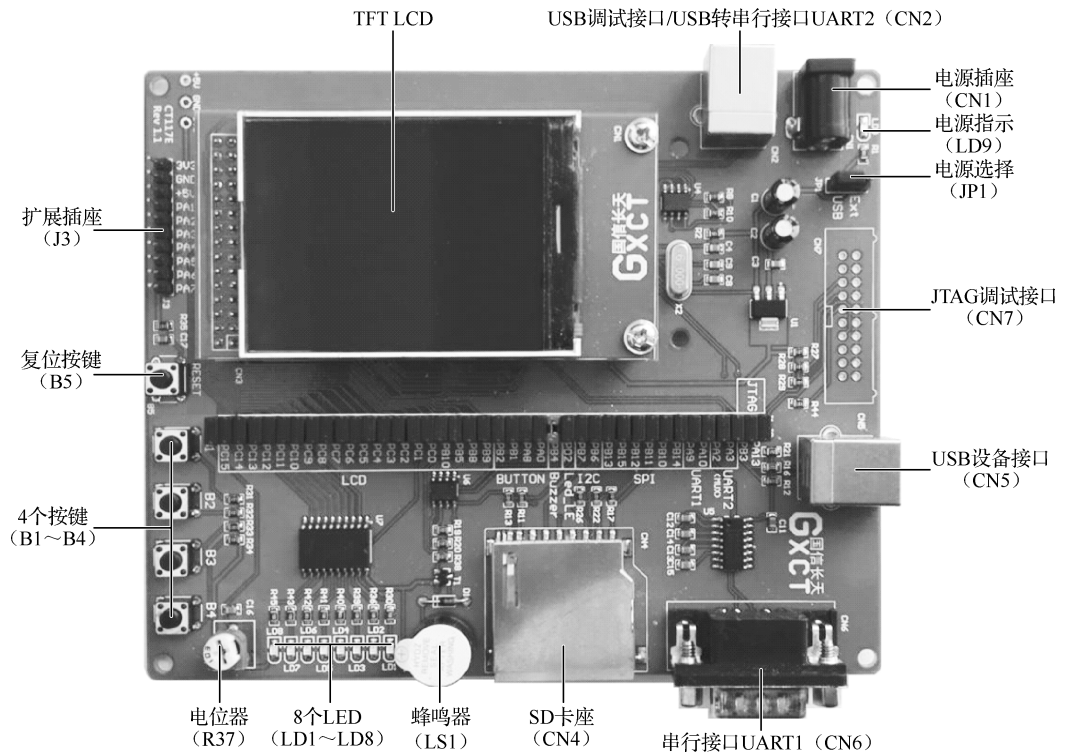



图 B.3　国信长天嵌入式竞赛实训平台实物图 2

国信长天嵌入式竞赛实训平台电路图如图 B.4 所示。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

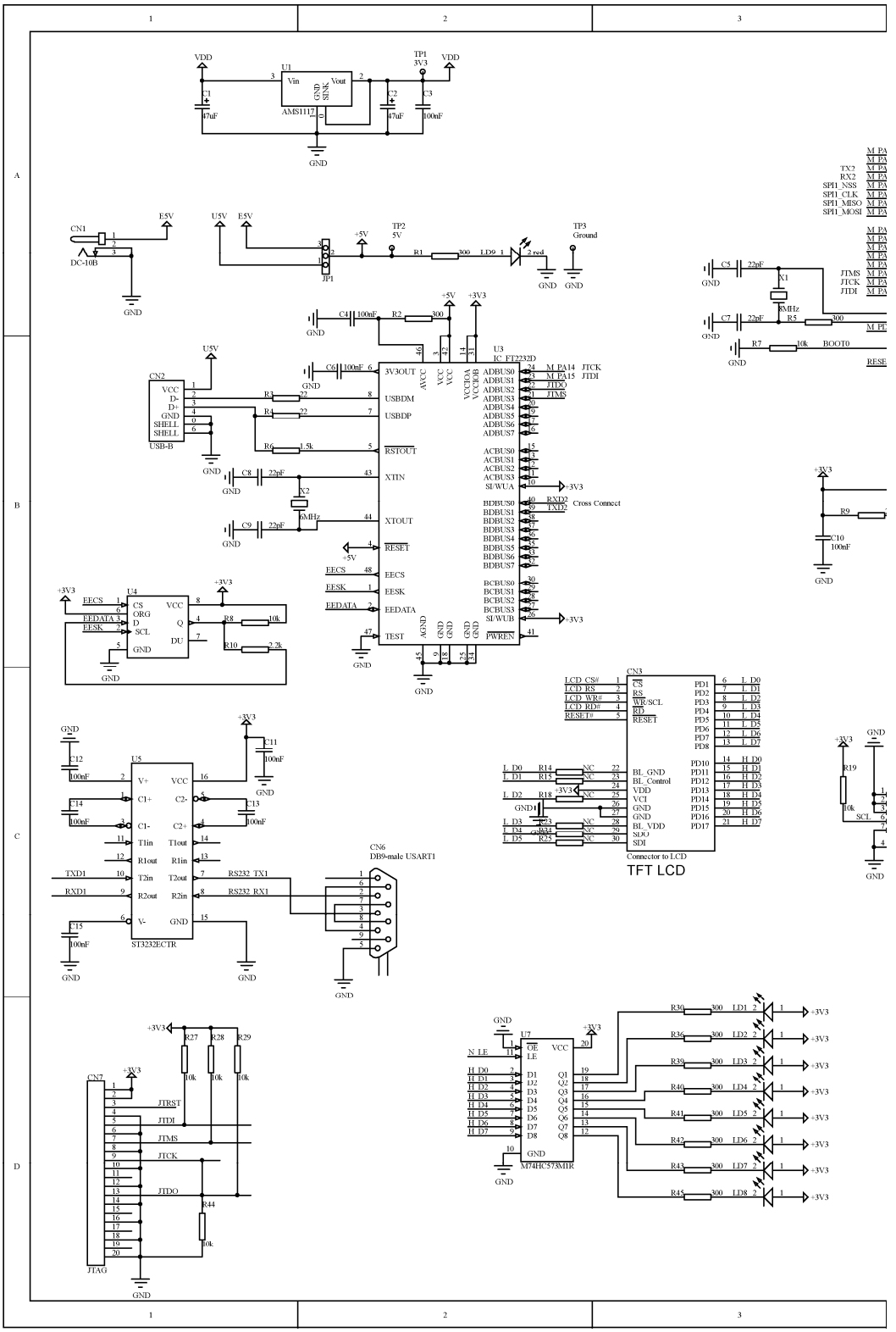
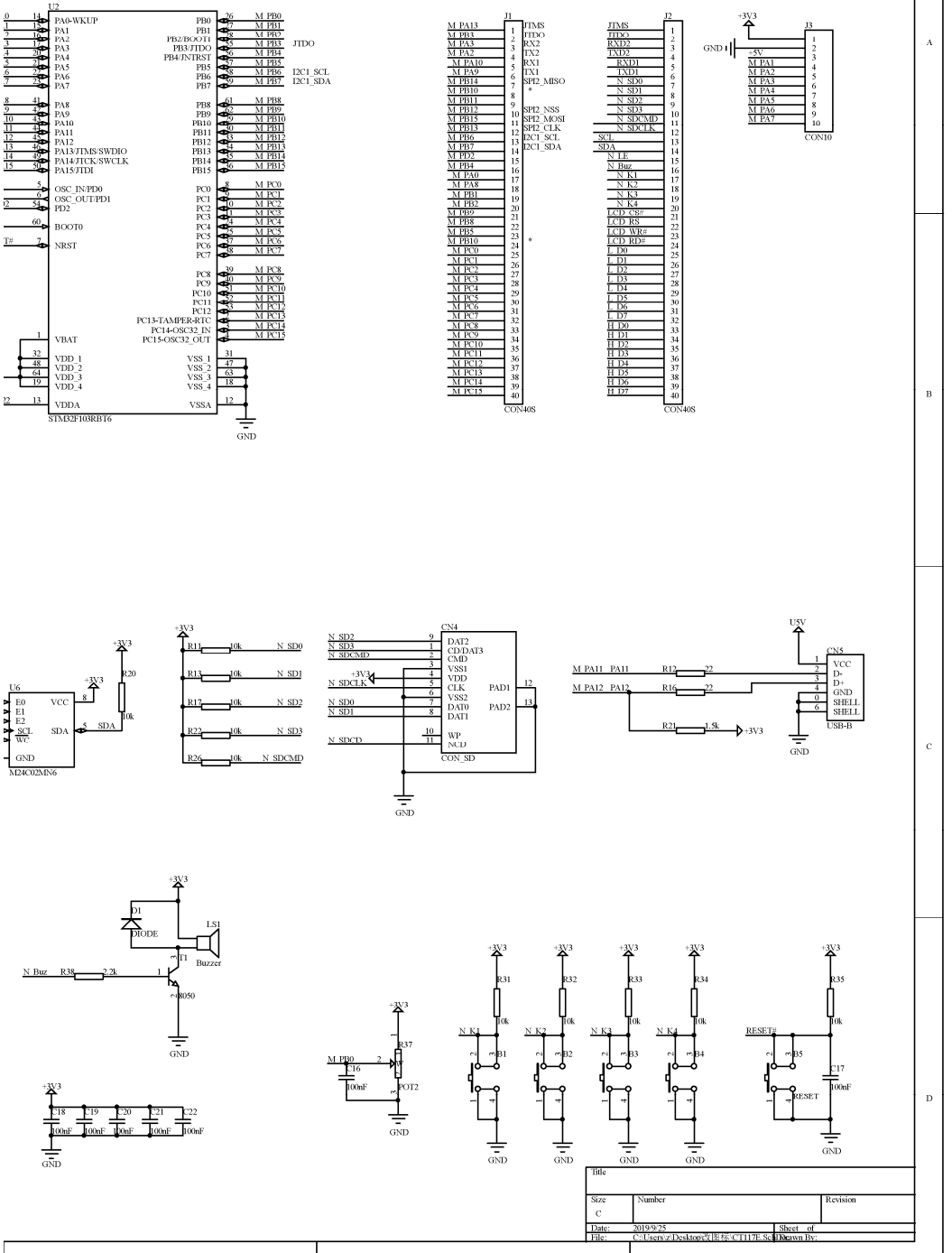



图 B.4　国信长天嵌入式



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY




竞赛实训平台电路图

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

# 附录 C　嵌入式资源扩展板

嵌入式资源扩展板由以下功能模块组成：

- 3 位八段数码管（DS1～DS3：共阴极静态显示）
- 8 个 ADC 按键（S1～S8，AKEY）
- 湿度传感器（U10：DHT11，HDO）
- 温度传感器（U9：DS18B20，TDO）
- 三轴加速度传感器（U8：LIS302DL）
- 光敏电阻（R47：10K，模拟输出 TRAO，数字输出 TRDO）
- 2 路模拟电压输出（AO1 和 AO2：输出电压范围 0～3.3V）
- 2 路脉冲信号输出（PLUS1 和 PLUS2：频率可调范围 100Hz～20kHz）
- 2 路 PWM 信号输出（PWM1 和 PWM2：固定频率，占空比可调范围：1%～99%）

嵌入式资源扩展板实物图如图 C.1 所示。

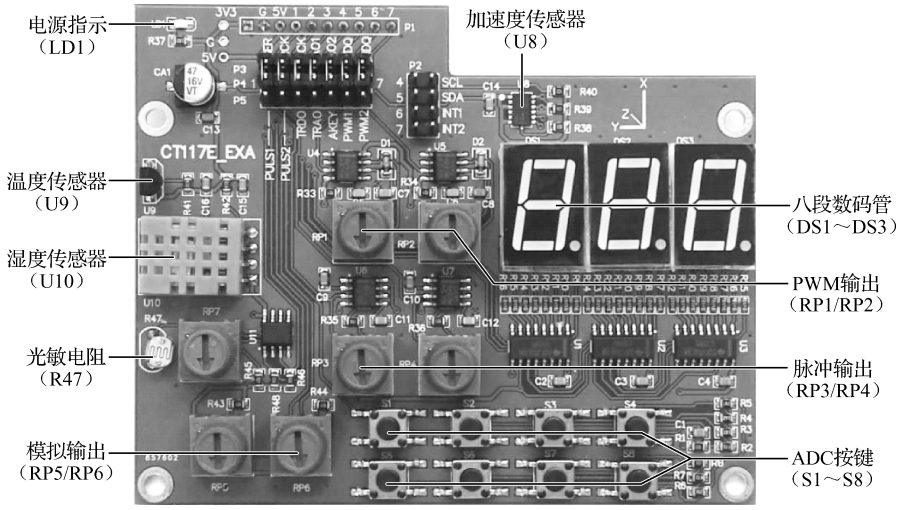



图 C.1　嵌入式资源扩展板实物图

嵌入式资源扩展板方框图如图 C.2 所示。

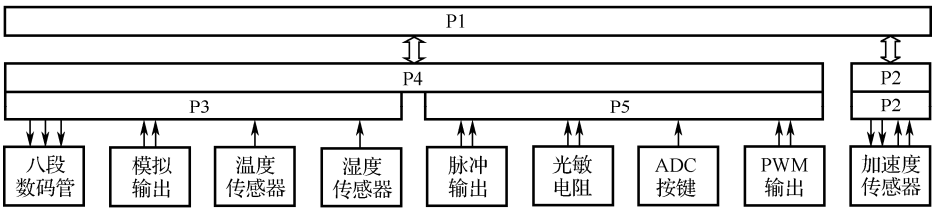



图 C.2　嵌入式资源扩展板方框图

嵌入式资源扩展板设备连接关系如表 C.1 所示。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

表 C.1　嵌入式资源扩展板设备连接关系

| 引　脚 | 功　能 | 说　明 | 引　脚 | 功　能 | 引　脚 | 功　能 | 说　明 |
|---|---|---|---|---|---|---|---|
| P1.1 | VDD | 3.3V 电源 | – | – | – | – | – |
| P1.2 | GND | 电源地 | – | – | – | – | – |
| P1.3 | VCC | 5V 电源 | – | – | – | – | – |
| P1.4 | PA1 | ADC_IN1/TIM2_CH2 | – | – | – | – | – |
| P1.5 | PA2 | ADC_IN2/TIM2_CH3 | – | – | – | – | – |
| P1.6 | PA3 | ADC_IN3/TIM2_CH4 | – | – | – | – | – |
| P1.7 | PA4 | ADC_IN4 | P2.1 | PA4 | P2.2 | SCL | 加速度传感器时钟 |
| P1.8 | PA5 | ADC_IN5 | P2.3 | PA5 | P2.4 | SDA | 加速度传感器数据 |
| P1.9 | PA6 | ADC_IN6/TIM3_CH1 | P2.5 | PA6 | P2.6 | INT1 | 加速度传感器中断 1 |
| P1.10 | PA7 | ADC_IN7/TIM3_CH2 | P2.7 | PA7 | P2.8 | INT2 | 加速度传感器中断 2 |
| P3.1 | SER | 数码管串行数据 | P4.1 | PA1 | P5.1 | PLUS1 | 脉冲信号输出 1 |
| P3.2 | RCK | 数码管数据锁存时钟 | P4.2 | PA2 | P5.2 | PLUS2 | 脉冲信号输出 2 |
| P3.3 | SCK | 数码管数据移位时钟 | P4.3 | PA3 | P5.3 | TRDO | 光敏电阻数字输出 |
| P3.4 | AO1 | 模拟电压输出 1 | P4.4 | PA4 | P5.4 | TRAO | 光敏电阻模拟输出 |
| P3.5 | AO2 | 模拟电压输出 2 | P4.5 | PA5 | P5.5 | AKEY | ADC 按键输出 |
| P3.6 | TDO | 温度传感器数据输出 | P4.6 | PA6 | P5.6 | PWM1 | PWM 信号输出 1 |
| P3.7 | HDO | 湿度传感器数据输出 | P4.7 | PA7 | P5.7 | PWM2 | PWM 信号输出 2 |

注：① P4.1～P4.3（PA1～PA3）可以通过短路块连接 P3.1～P3.3 或 P5.1～P5.3，但两者不能同时连接，即数码管和脉冲信号输出/光敏电阻数字输出不能同时使用。

② P4.4～P4.7 或 P2.1、P2.3、P2.5、P2.7（PA4～PA7）可以通过短路块连接 P3.4～P3.7 或 P5.4～P5.7 或 P2.2、P2.4、P2.6、P2.8，但三者不能同时连接，即模拟电压输出/温度传感器数据输出/湿度传感器数据输出和光敏电阻模拟输出/ADC 按键输出/PWM 信号输出以及加速度传感器不能同时使用。

③ 在扩展板上使用 PA2 和 PA3（如使用数码管）时，实训平台上的 UART2（UART 转 USB）无法使用。

④ 如果出现冲突或者不能满足设计要求时，可以用导线连接其他不用引脚。

嵌入式资源扩展板电路图如图 C.3 所示。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

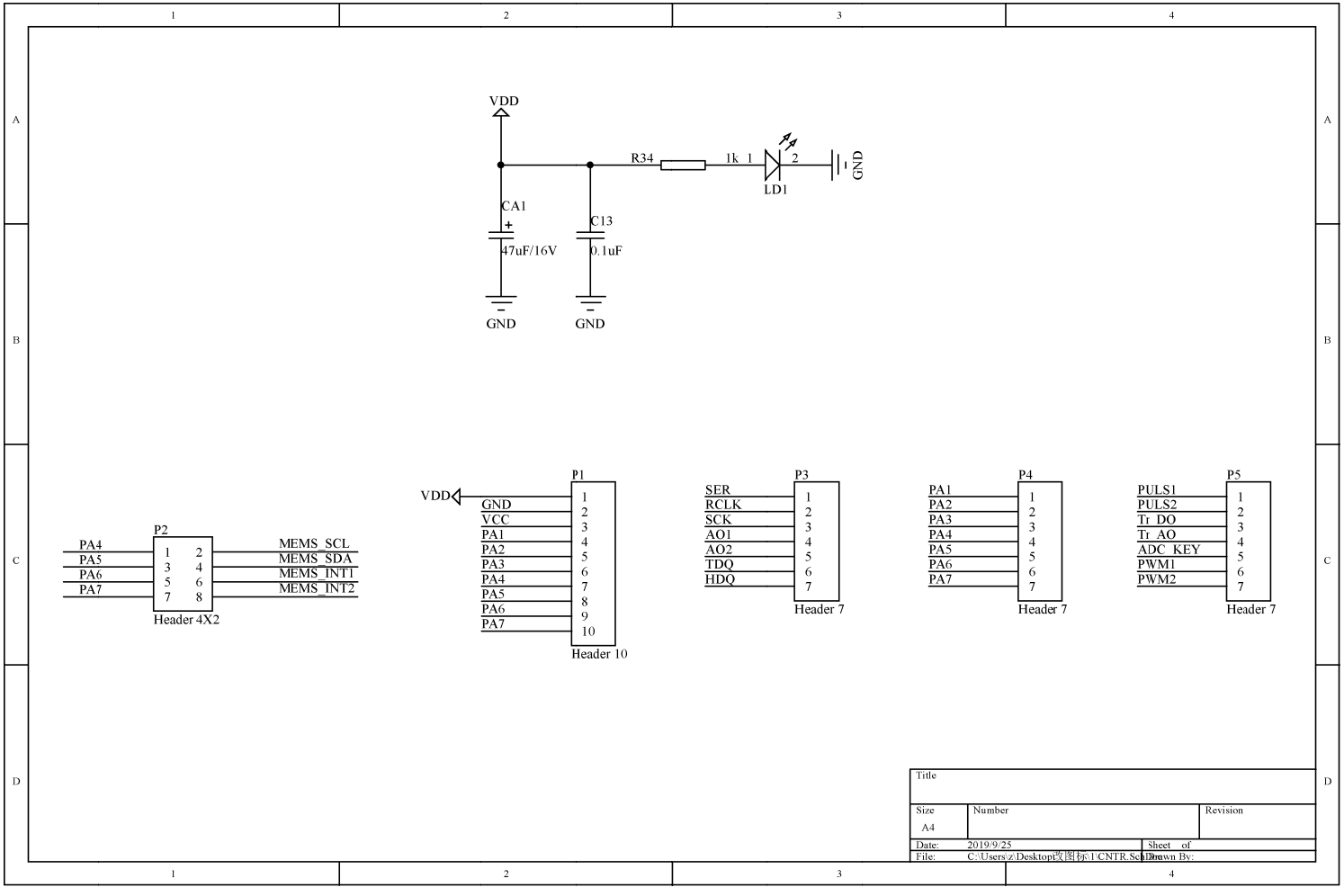



图 C.3　嵌入式资源扩展板电路图

电子工业出版社 PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

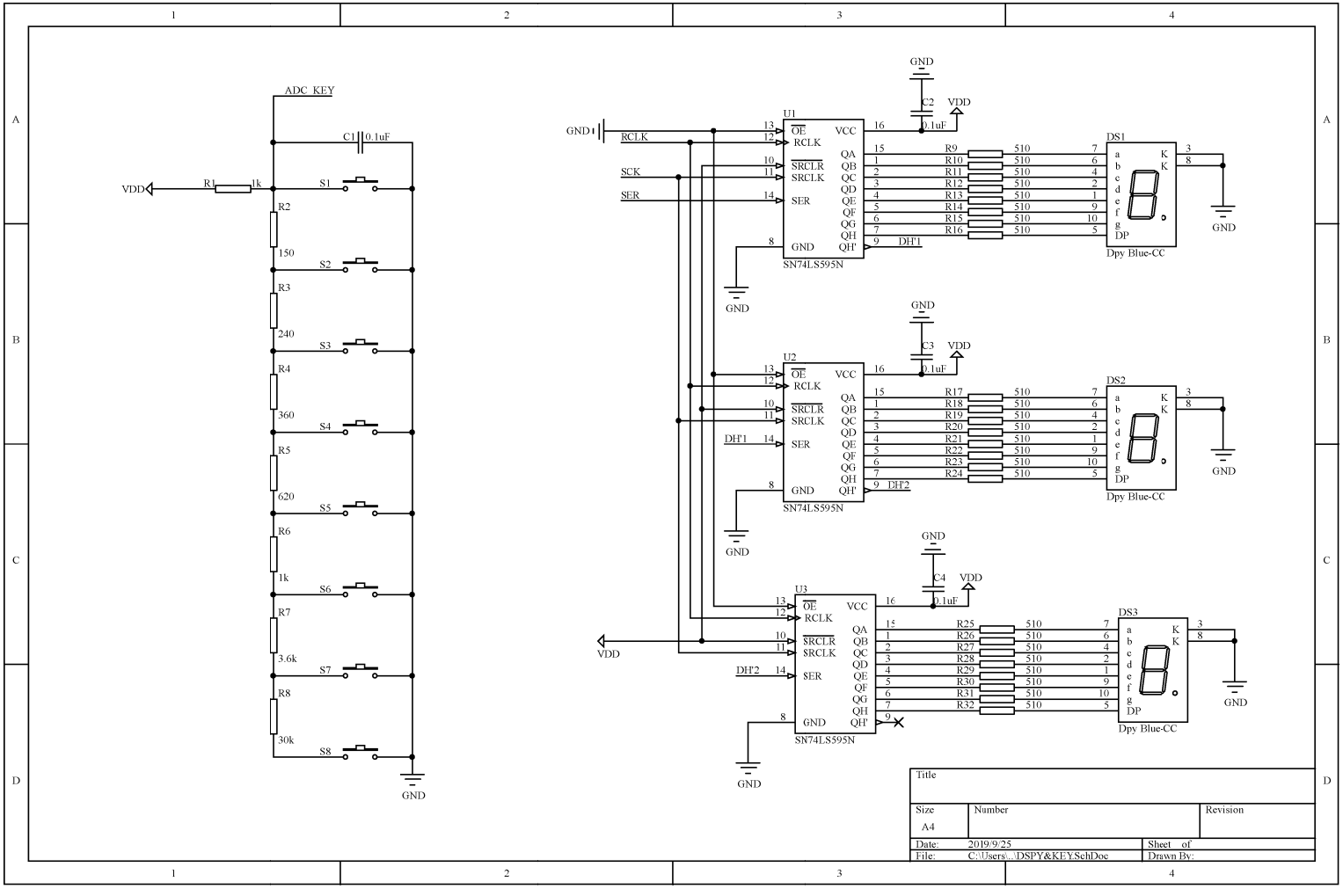



图 C.3　嵌入式资源扩展板电路图（续）

電子工業出版社 PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

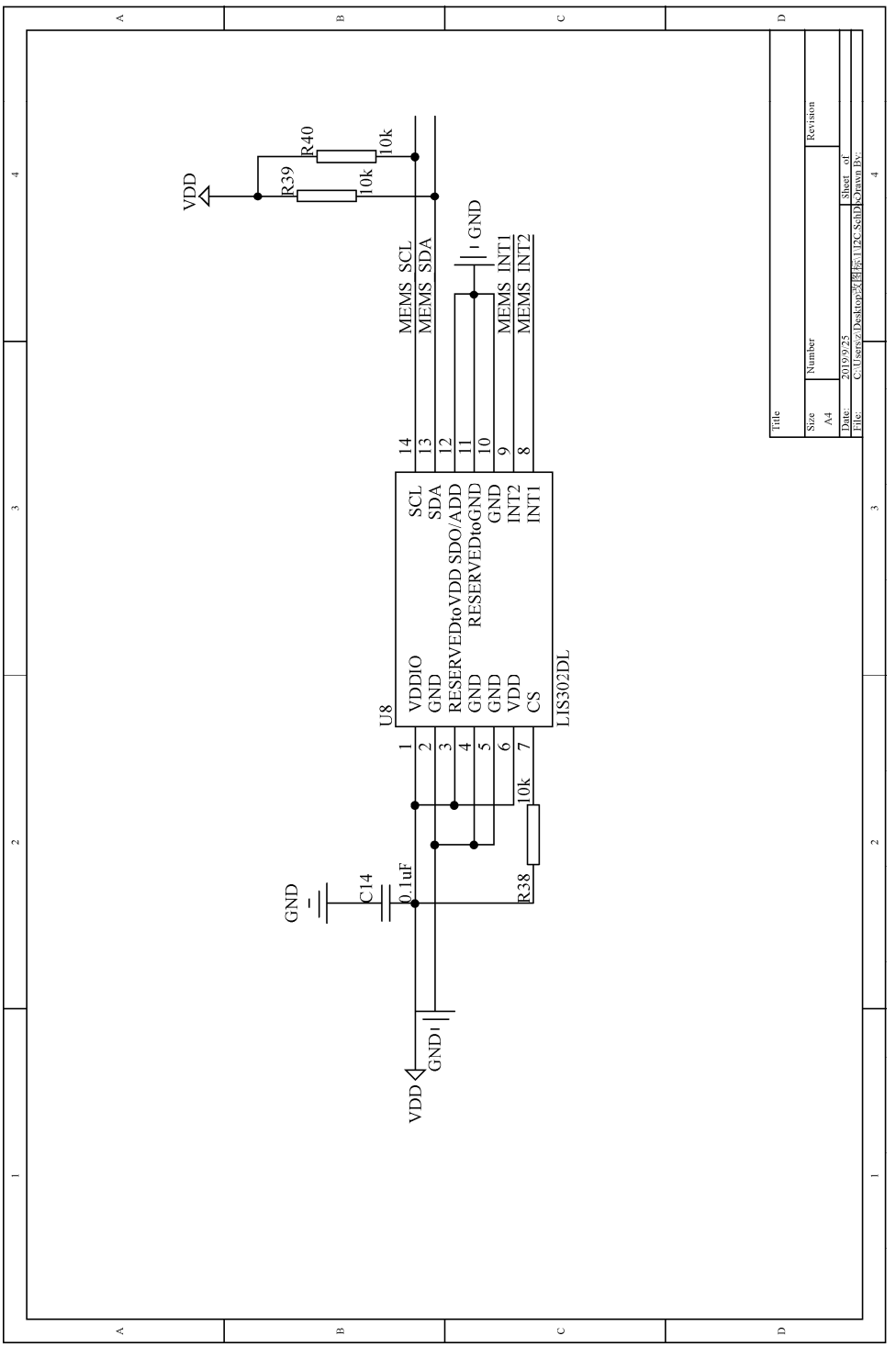



图 C.3　嵌入式资源扩展板电路图（续）



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

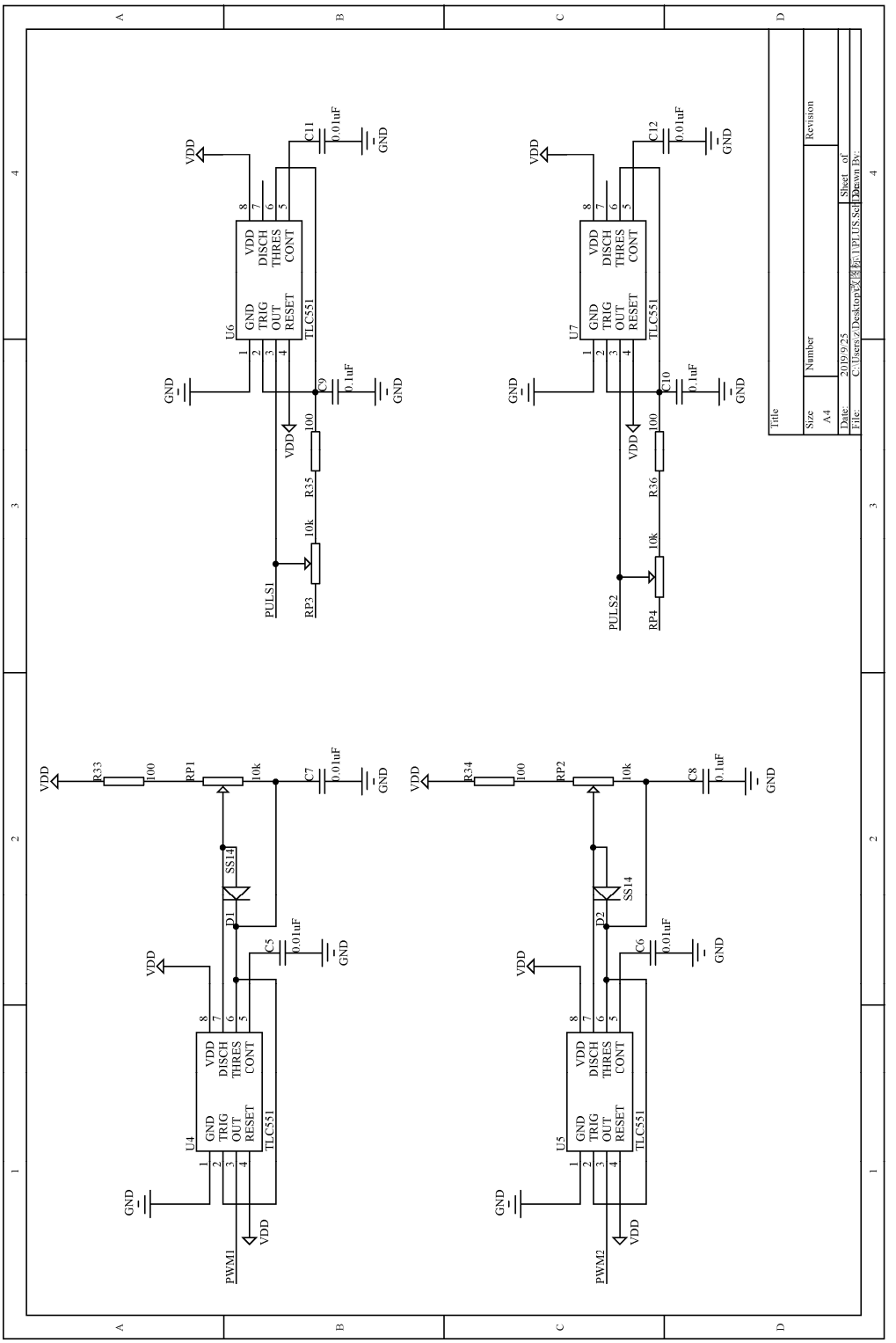



图 C.3　嵌入式资源扩展板电路图（续）

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

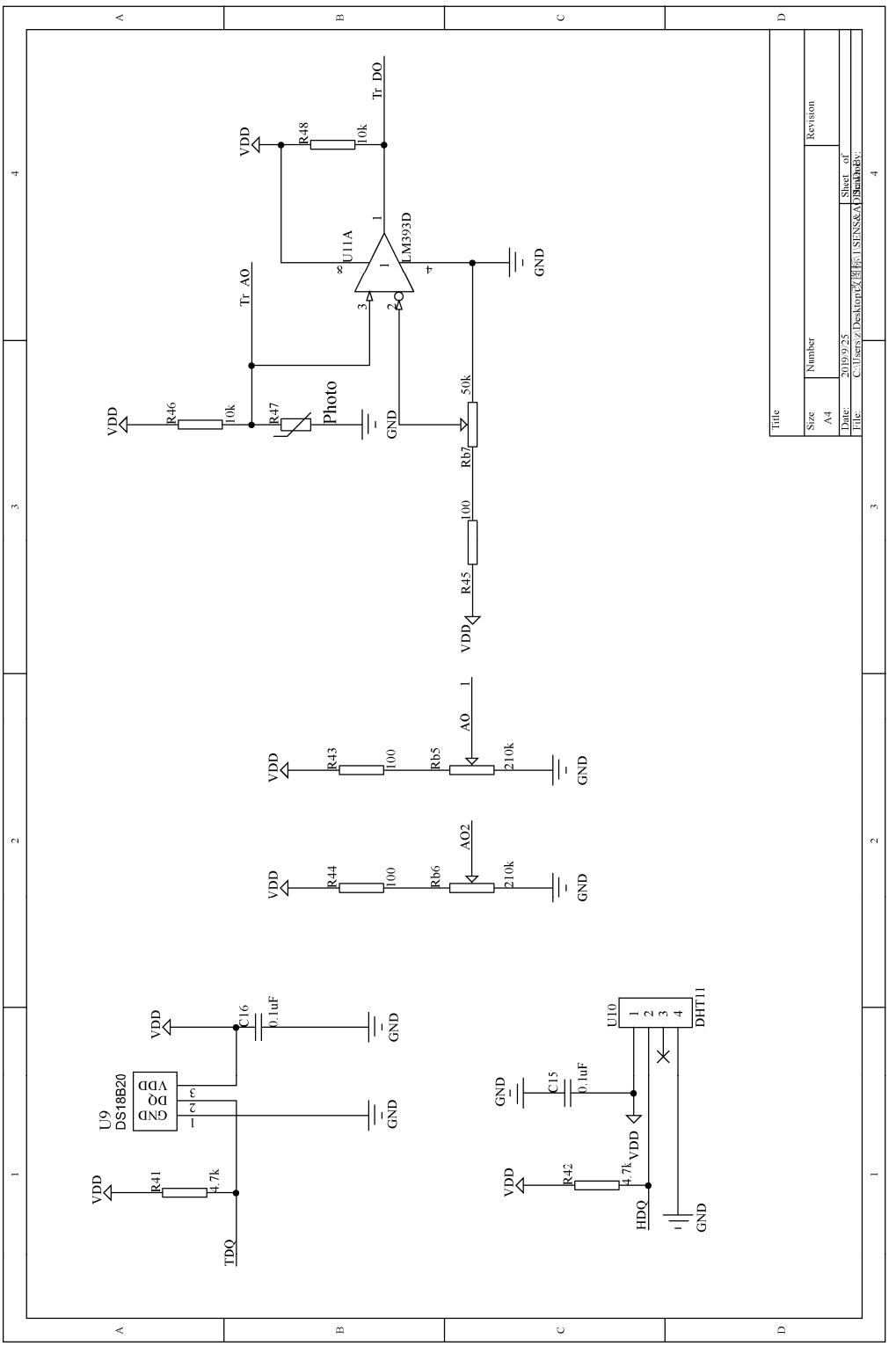



图 C.3　嵌入式资源扩展板电路图（续）



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

# 附录 D　单片机设计与开发项目竞赛大纲

**1）参赛资格**

具有正式学籍的在校全日制本科及高职高专学生（以报名时状态为准）。

**2）竞赛用时**

预赛时长：5 小时，决赛时长：5 小时。

**3）竞赛形式**

个人赛，预赛和决赛均采用封闭和限时方式进行。

选手机器通过局域网连接到各个考场的竞赛服务器。选手答题过程中无法访问互联网，以“服务器–浏览器”方式发放试题、回收选手答案。

备注：选手提交到服务器的工程文件是成绩复查的依据，须确保工程文件提交完整。

**4）试题形式**

竞赛试题由客观题和基于统一硬件平台的程序设计与调试试题两部分组成。

（1）客观题

- 选择题：选手根据题目描述，选择若干个答案
- 填空题：题目描述一个具有确定解的问题，选手根据题目要求填写唯一答案

（2）硬件程序设计试题

- 硬件平台：IAP15F2K61S2 单片机，国信长天单片机竞赛实训平台，www.gxct.net
- 试题形式：参赛选手在规定时间内，基于国信长天单片机竞赛实训平台，按照试题要求使用 C 语言或汇编语言完成设计开发与调试任务

**5）赛场设备设施**

（1）硬件设施

- 万用表：数字万用表，具备直流电压、电流、电阻测量和通断测试功能
- 计算机：X86 兼容机器，内存不小于 1G，硬盘不小于 60G，操作系统：Windows XP/7

（2）软件预装

- Keil C51 集成开发环境
- STC-ISP v6.86（或以上版本）程序下载软件
- 单片机竞赛实训平台驱动程序

**6）试题涉及的基础知识**

- C 程序设计基础知识
- 模拟/数字电子技术基础
- MCS-51 单片机基础
- MCS-51 单片机程序开发与调试

**7）分值比例**

客观题：30%，基于硬件平台的程序设计与调试：70%。

**8）评分**

客观题：答案唯一，每题只有 0 分或满分，全部机器阅卷；硬件平台程序设计与调试试题：专家组根据参赛选手功能完成情况，依据评分细则标准打分。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

# 附录 E　嵌入式设计与开发项目竞赛大纲

**1）参赛资格**

具有正式学籍的在校全日制本科及高职高专学生（以报名时状态为准）。

**2）竞赛用时**

预赛时长：5 小时，决赛时长：5 小时。

**3）竞赛形式**

个人赛、预赛和决赛均采用封闭和限时方式进行。

选手机器通过局域网连接到各个考场的竞赛服务器。选手答题过程中无法访问互联网，以“服务器–浏览器”方式发放试题、回收选手答案。

备注：选手提交到服务器的工程文件是成绩复查的依据，须确保工程文件提交完整。

**4）试题形式**

竞赛试题由客观题和基于统一硬件平台的程序设计与调试试题两部分组成。

（1）客观题

- 选择题：选手根据题目描述，选择若干个答案
- 填空题：题目描述一个具有确定解的问题，选手根据题目要求填写唯一答案

（2）硬件程序设计试题

- 硬件平台：STM32F103RBT6（ARM Cortex M3），国信长天嵌入式竞赛实训平台，嵌入式资源扩展板（扩展传感器和执行机构，www.gxct.net）
- 试题形式：参赛选手在规定时间内，基于国信长天嵌入式竞赛实训平台和嵌入式资源扩展板，按照试题要求使用 C 语言或汇编语言完成设计开发与调试任务

**5）赛场设备设施**

（1）硬件设施

- 万用表：数字万用表，具备直流电压、电流、电阻测量和通断测试功能
- 计算机：X86 兼容机器，内存不小于 1GB，硬盘不小于 60GB，操作系统：Windows XP/7
- 双踪示波器

（2）软件预装

- Keil MDK 集成开发环境（v4.11～v4.74 版本可用）
- Colink MDK 插件
- STM32 SPLib v3.5
- 嵌入式竞赛实训平台驱动程序（FT2232 驱动程序）

备注：5.0 级以上版本的 Keil 集成开发环境与 Coolink 调试器插件存在兼容性问题，不建议作为竞赛开发环境安装使用。

**6）试题涉及的基础知识**

- C 程序设计基础知识
- 模拟/数字电子技术基础
- ARM Cortex M3 软件编程与调试
- 基于 STM32 处理器的开发应用



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

7）分值比例

客观题：30%，基于硬件平台的程序设计与调试：70%。

8）评分

客观题：答案唯一，每题只有 0 分或满分，全部机器阅卷；硬件平台程序设计与调试试题：专家组根据参赛选手功能完成情况，依据评分细则标准打分。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

# 参 考 文 献


[1] STMicroelectronics．STM32F10xxx 参考手册．2010

[2] STMicroelectronics．STM32F10xxx Reference Manual（RM0008）．2011

[3] 李宁．ARM MCU 开发工具 MDK 使用入门．北京：北京航空航天大学出版社，2012

[4] 郭书军，王玉花．ARM Cortex-M3 系统设计与实现——STM32 基础篇，北京：电子工业出版社，2014

[5] 郭书军．ARM Cortex-M4 + WiFi MCU 应用指南——CC3200 IAR 基础篇，北京：电子工业出版社，2016

[6] 郭书军．ARM Cortex-M3 系统设计与实现——STM32 基础篇（第 2 版），北京：电子工业出版社，2018



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY